REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. **REPORT DATE** *(DD-MM-YYYY)* 28-02-2014	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* 01 Dec 2011 – 28 Feb 2014

4. TITLE AND SUBTITLE Novel Algorithm/Hardware Partnerships for Real-Time Nonlinear Control	**5a. CONTRACT NUMBER** FA9550-12-C-0035
	5b. GRANT NUMBER NA
	5c. PROGRAM ELEMENT NUMBER NA
6. AUTHOR(S) Yun Wang Ben G. Fitzpatrick	**5d. PROJECT NUMBER** NA
	5e. TASK NUMBER NA
	5f. WORK UNIT NUMBER NA
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stochastech Corporation dba Tempest Technologies 8939 S. Sepulveda Blvd., Suite 506 Los Angeles, CA 90045	**8. PERFORMING ORGANIZATION REPORT NUMBER** NA
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AIR FORCE RESEARCH LABORATORY AF OFFICE OF SCIENTIFIC RESEARCH 875 N. RANDOLPH ST. ROOM 3112 ARLINGTON VA 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The real-time implementation of controls in nonlinear systems remains one of the great challenges in applying advanced control technology. The work under this Phase II SBIR involves an innovation in computational nonlinear control that offers ground breaking potential for real-time control applications, making fully nonlinear problems solvable with the computational efficiency of linear problems.

15. SUBJECT TERMS
Nonlinear control, max-plus, FPGA, perimeter patrol,

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Fariba Fahroo
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	71	**19b. TELEPHONE NUMBER** *(include area code)* 703-696-8429

Novel Algorithm/Hardware Partnerships for Real-Time Nonlinear Control

Final Technical Report
ITEM No. 0001AF
Contract FA9550-12-C-0035
Submitted by
Yun Wang, Ph.D. / Principal Investigate
Tempest Technologies
28 February 2014

Abstract

The real-time implementation of controls in nonlinear systems remains one of the great challenges in applying advanced control technology. Often, linearization around a set point is the only practical approach, and many controllers implemented in hardware systems are simple proportional-integral-derivative (PID) feedback mechanisms. To apply Pontryagin's principle or Bellman's equation using conventional hardware and algorithms for high dimensional nonlinear systems requires more computing power than is realistic. The success of linear control theory, especially certainty equivalence and linear-quadratic-Gaussian (LQG) approaches, leads us to hope for additional gains from fully nonlinear controls. The work under this Phase II SBIR involves an innovation in computational nonlinear control that offers ground breaking potential for real-time control applications, making fully nonlinear problems solvable with the computational efficiency of linear problems.

This report details the development of integrated hardware-software solutions implementing max-plus arithmetic for efficient solution of nonlinear control problems. We have developed nonlinear controls for a complex perimeter patrol problem posed by Air Force Research Lab personnel. Our general nonlinear control software package is in beta testing.

TABLE OF CONTENTS

Abstract ii

1.0 Executive Summary 1

2.0 Nonlinear Optimal Control Problems 4

3.0 The Max-Plus Arithmetic 10

4.0 Section Nonlinear Control and The Max-Plus Algebra 15

5.0 A Max-Plus Approach to Stochastic Control 21

6.0 Max-Plus Algebra and Hardware/Software Considerations 27

6.1 The Xilinx FPGA ML605. 28

6.2 The max-plus concept in FPGA 31

6.3 Running the system 34

6.4 Max-Plus FPGA MVM computation 38

6.5 Timing results 42

7.0 The Linear Quadratic Regulator 43

8.0 Perimeter Patrol Planning – An Example 47

8.1 Perimeter Patrol Problem Definition 49

8.2 A Single UAV 54

8.3 Two UAVs 60

8.4 Three UAVs 63

9.0 Conclusions 64

10.0 References 65

LIST OF FIGURES

Figure 3.1 Linear and Quadratic Max-Plus Finite Element Approximations 14
Figure 3.2 The supporting lines and the resulting approximation of max-plus Legendre elements 14
Figure 4.1 Mixed finite element test and basis functions trapping a function to be approximated 19
Figure 6.1 Concept block diagram Tempest's Max-Plus MVM FPGA 28
Figure 6.2 Xilinx ML605 FPGA board 29
Figure 6.3 Xilinx ML605 FPGA components 30
Figure 6.4 Look-up table (LUT) cells, multiply-add (MUX) cells, and fabric structure 31
Figure 6.5 First row of max-plus matrix multiply 32
Figure 6.6 Max-Plus Iteration Core organization 33
Figure 6.7 Architecture of the full host-coprocessor system 34
Figure 6.8 Comparison of host processor (MEX) and FPGA max-plus matrix iteration speed ... 43
Figure 7.1 Analytically computed B and M matrices for a 1-D LQR problem: left panel shows the B matrix and right panel shows the M matrix 45
Figure 7.2 Numerically computed B and M matrices for a 1-D LQR problem: left panel shows the B matrix and right panel shows the M matrix 45
Figure 7.3 Analytic and numeric value functions, together with their difference 46
Figure 7.4 Approximation error heat maps for coarse and fine grid approximations 47
Figure 8.1 Perimeter patrol illustration 49
Figure 8.2 An example information gain function 52
Figure 8.3 Max-Plus Probability Solutions Parameter Sweep 55
Figure 8.4 Aggregate Solution vs. Standard Solution 55
Figure 8.5 Max-Plus Probability Solution vs. Standard Solution 56
Figure 8.6 Aggregate Solution and Standard Solution Dwells 57
Figure 8.7 Max-Plus Probability Solution and Standard Solution Dwells 57
Figure 8.8 Dwell Behavior, Left d=5 Right d=6 58
Figure 8.9 Aggregate Solution and Standard Solution Waits 58
Figure 8.10 Max-Plus Probability Solution and Standard Solution Waits 59
Figure 8.11 Left panel shows the optimal cost of the Standard Solution for a sampling of initial states under a variety of dwell/wait time upper bounds. Right panel shows cost as a function of the upper bounds on dwells (top right) and waits (bottom right) 59
Figure 8.12 Convergence of Max-Plus Probability Solutions on the Standard Solution's performance measure 60
Figure 8.13 Left panel shows the optimal cost of the Standard Solution for a sampling of initial states under a variety of dwell/wait time upper bounds. Right panel shows cost as a function of the upper bounds on dwells (top right) and waits (bottom right) 60
Figure 8.14 Dwell Dependence of Max-Plus Probability Solutions on the Standard Solution's performance measure 61

Figure 8.15 Performance Stable with Dwell Behavior Changing Maximum Dwell and Wait Increasing left (d=5, w=3) to right (d=8, w=6) (top: Optimal Performance bottom: Dwells Histograms) 61
Figure 8.16 Max-Plus Probability Solution vs. Aggregate Solution 62
Figure 8.17 Max-Plus Probability Solution and Aggregate Solution Dwells........ 62
Figure 8.18 Max-Plus Probability Solution and Aggregate Solution Waits........ 63
Figure 8.19 Max-Plus Probability Solution Performance........ 63
Figure 8.20 Comparison of Integration Methods........ 64

1.0 Executive Summary

Control problems arise in response to the need to achieve desired performance. Applications range from satellite orbit insertion to high energy laser tracking and pointing to design of influence operations, and technologies on which the Air Force relies commonly require careful attention to control objectives. Common control problems include maintaining a specific range of operation (stabilization) or driving the system from one state to another (steering), but many applications naturally produce optimization problems, in which we seek maximum information, minimum resource expenditure, minimum time to task completion, or maximum profit.

The mathematical theory of optimal control operates with two fundament objects: a mathematical model of the system's dynamics, and a performance criterion or objective we seek to optimize. Models of system dynamics range from simple discrete dynamical systems to ordinary and partial differential equations to Markov chains and processes and stochastic differential equations. In many cases models are derived from physical principles (Newton's laws, lift and drag, conservation of mass and momentum), mechanistic considerations (susceptibility and contact rates in epidemics), while empirical model designs are useful in many cases as well.

The primary challenges we have attacked in this effort involve nonlinearity and uncertainty. The control of linear systems is well understood, and a wide variety of techniques exist for designing controllers for linear systems. Many systems of interest in demanding aerospace and defense applications exhibit nonlinear dynamics. In some cases, one can apply controllers designed through linear techniques that perform well under a reasonable range of operating conditions. When pushing the envelope of performance, operating a system in a nonlinear regime may be crucial. Toward that end, we have focused on a set of computational techniques suitable for approaching a broad spectrum of nonlinear problems.

Stochasticity represents additional challenges in nonlinear systems. The difficulties of designing control strategies are compounded by uncertainties in dynamics and measurements. Linearization, coupled with the separation principle, provides the most commonly used tool for stochastic control. The tools we have developed are well-suited to handling uncertainty in a very different way.

Computation remains a key complication for nonlinear control. Linearizing the dynamics around a set point, trajectory, or equilibrium of interest, is often the only satisfactory approach. Some problems admit a transformation of variables that will change the system into a linear one, but the requirements on the dynamical system limit this to a few specific applications. Gain scheduling algorithms are often coupled with linearizations, applying the appropriate linear feedback for a given operating regime. In some cases, these controllers are augmented with adaptive schemes to adjust the plant model on-line to enhance the linearized feedback gain controls. In recent years, H^{∞} techniques have been applied to improve robustness against perturbations. In nearly every case, however, these approaches involve linear approximations.

Nonlinear approaches are typically based on one of two technologies: Pontryagin's maximum principle and dynamic programming. The former is essentially a Lagrange multiplier approach to control as an abstract constrained optimization. Generally Pontryagin's solution is an open loop control strategy. A further complication is that the two-point boundary value problem that results from its application can be a numerical challenge.

Dynamic programming, on the other hand, can lead to feedback controllers, but its computational cost has proven quite high. Most numerical approaches arise from consideration of the Hamilton-Jacobi-Bellman PDE that is derived from the dynamic programming principle. The value function that solves this PDE provides access to the optimal control in terms of the state. Here we apply numerical approximation directly to the dynamic programming principle, bypassing the PDE approach. Our numerical algorithms rely on mathematical techniques designed specifically for a key feature of dynamic programming: we redefine arithmetic in order to make the dynamic programming principle a linear equation. That arithmetic framework is the max-plus algebra. This approach allows us to solve the dynamic programming equation as a linear problem in a different computational paradigm, thereby computing nonlinear controls economically.

Dynamic programming also provides a paradigm for the treatment of stochasticity, a problem that is quite difficult for Pontryagin's maximum principle to handle. Using max-plus arithmetic, we have an additional modeling tool for uncertainty in max-plus probability. Here we consider both traditional and max-plus stochastic approaches to control of uncertain systems.

In this report, we detail our work in developing max-plus methods for deterministic and stochastic control, along with our hardware/software partnerships for efficient computation.

Personnel involved in the effort include

1. Yun Wang, PhD, Principal Investigator
2. Ben G. Fitzpatrick, PhD, Senior Scientist
3. Matthew Laffin, MS, Research Assistant
4. Kristin Holmbeck-Cannell, Research Assistant
5. Gang "George" Yin, PhD, Senior Scientist (WSU)
6. Araz Hashemi, MS, Research Assistant (WSU)
7. Le Yi Wang, PhD, Senior Scientist (WSU)

Publications resulting from this effort are as follows:

1. Hashemi, A., B. Fitzpatrick, L.Y. Wang, and G. Yin "Robust noise attenuation under stochastic noises and worst-case unmodeled dynamics," to appear in *International Journal of Systems Science*.
2. B. G. Fitzpatrick, K. Holmbeck-Cannell, M. Laffin, and Y. Wang. "Optimal Perimeter Patrol via Max-Plus Probability," to appear in *2014 Proc ACC*.
3. Hashemi, A. "Adaptive stochastic systems: estimation, filtering, and noise attenuation," PhD dissertation, Wayne State University Department of Mathematics.

2.0 Nonlinear Optimal Control Problems

We begin the discussion of our max-plus technology with an overview of nonlinear control problems. Generally speaking, the two main ingredients of a control system are the dynamic model and the control objective. With respect to the dynamic model, control problems take many forms, but there are two primary types of categorizations: discrete vs. continuous dynamics and deterministic vs. stochastic dynamics.

A deterministic dynamical system is often modeled with a difference equation, such as

$$x(t+1) = f(x(t), u(t)), \quad x(t_0) = x_0 \tag{2.1}$$

in which the system under control propagates at discrete time steps. The variable x denotes the state of the system, while the variable u denotes the control input. The system is initially in the state at time , and we wish to influence the system to behave in a desired manner by choosing the control over a time interval Control objectives are typically set as running costs that are measured over the entire period of operation and final time costs that penalize or reward the ultimate state:

$$J(u, x_0, t_0) = \sum_{k=0}^{N-1} L(x(t_0+k), u(t_0+k)) + \Phi(x(t_f)), \tag{2.2}$$

a functional that is to be maximized over the vector of possible control actions. A number of possible solution approaches may be taken, from “brute force” multivariate optimization to dynamic programming, the latter providing the basis of the work described herein.

Bellman’s principle of dynamic programming relies on the value function, which is defined as

$$V(t,x) = \max_{u(\bullet)} \{J(u,x,t)\}, \tag{2.3}$$

which provides the optimal value of the cost functional under initial condition x at time t. The principle of dynamic programming is this: whatever the initial state, the remaining decisions must be optimal for the state that results from the first control decision. Mathematically, this statement becomes

$$V(t,x) = \max_{u(t)} \{ L(x,u) + V(t+1, f(x(t),u(t))) \}, \tag{2.4}$$

in which the value function inside the max represents the optimality of the future control after that initial decision. With (2.4), the control problem is reduced to a backward iteration from the final time value function of $V(t_f, x) = \Phi(x)$. At each time step and for each state, the optimal control at that time is determined by this recursion. Thus, the efficient computation of the value function is a key issue for the dynamic programming approach to optimal control.

For continuous time problems, we formulate the dynamics with a differential equation, which we take to be of the form

$$\dot{x} = f(x,u), \quad x(t_0) = x_0, \tag{2.5}$$

modeling the system of interest. The second ingredient is the control objective given by

$$J(u, x_0, t_0) = \int_{t_0}^{t_f} L(x(t), u(t))dt + \Phi(x(t_f)), \tag{2.6}$$

which is to be maximized over the set of admissible control functions, $u \in U(t_0, t_f) \subset L^2(t_0, t_f)$.

Solution of this problem typically takes one of two routes: constrained optimization relying on Pontryagin's maximum principle, or dynamic programming relying on the Bellman equation. The use of Pontryagin's principle, which is essentially the application of Lagrange multiplier methods for the constrained optimization problem of maximizing the objective in the presence of the dynamical system model constraint, continues to prove very difficult for real-time computation. The approach relies on the necessary condition that, if (x^*, u^*) denotes the optimal system state and control input pair, then they satisfy the equations

$$\begin{aligned} \frac{dx^*}{dt} &= f(x^*(t),u^*(t)), \quad x^*(t_0) = x_0 \\ \frac{d\lambda^*}{dt} &= -\frac{\partial f}{\partial x}(x^*(t),u^*(t))\lambda^* - \frac{\partial L}{\partial x}(x^*(t),u^*(t)), \quad \lambda^*(t_f) = 0 \end{aligned}, \tag{2.7}$$

a two-point boundary value problem. The function u^* is the maximizer of the Hamiltonian at each time:

$$u^*(t) = \arg\max_u \left(f(x^*(t),u(t))^T \lambda^*(t) + L(x^*(t),u(t)) \right) \text{ for each } t. \tag{2.8}$$

The problem, then, is to solve the pair of differential equations, one of which runs forwards in time, the other of which runs backwards, while at each time maximizing the Hamiltonian to determine the optimal control value. Numerical methods for this problem range from the simple shooting method to more sophisticated spectral and pseudospectral methods. A drawback to this method is that the resulting control is purely open loop: the control function is determined as a function of time and not state.

Applying dynamic programming in the continuous time case again requires the value function, which is defined in a manner similar to the discrete situation:

$$V(t_0, x_0) = \max_{u \in U} \left(J(u, x_0, t_0) \right). \tag{2.9}$$

The idea of dynamic programming in the continuous setting is that the optimal cost over a time period [t, t_f] comprises the optimal cost over a short subinterval [t, s] together with the optimal cost from s to t_f. To formalize this concept, we first denote the solution of the differential equation by $x(\bullet; t_0, x_0, u)$. Bellman's equation is then

$$V(t, y) = \max \left\{ \int_t^s L(x(\tau; t, y, u), u(\tau))d\tau + V(s, x(s; t, y, u)) \right\}, \tag{2.10}$$

in which s and t are times satisfying $t < s$. We include the dependence of the state trajectory on the initial time, initial state, and control. Dynamic programming in continuous time backs up from the final time to the initial time by repeatedly solving short time optimal control problems.

In continuous time, Bellman's equation leads to the Hamilton-Jacobi-Bellman partial differential equation (HJB PDE) characterization of the value function. We consider

$$\begin{aligned}
0 &= \max\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + V(s,x(s;t,y,u))\right\} - V(t,y) \\
&= \max\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + V(s,x(s;t,y,u)) - V(s,y)\right\} + V(s,y) - V(t,y)\,. \qquad (2.11) \\
&\approx \max\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \nabla V(s,y)\big(x(s;t,y,u)-y\big)\right\} + V(s,y) - V(t,y)
\end{aligned}$$

Dividing by (s-t), we have

$$\begin{aligned}
0 &= \max\left\{\frac{1}{s-t}\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \nabla V(s,y)\frac{x(s;t,y,u)-y}{s-t}\right\} + \frac{V(s,y)-V(t,y)}{s-t} \\
&\approx \max\left\{L(y,t) + \nabla V(t,y)\frac{dx}{dt}\right\} + \frac{\partial V}{\partial t} \qquad , \qquad (2.12) \\
&= \max\left\{L(y,t) + \nabla V(t,y)\, f(x,u)\right\} + \frac{\partial V}{\partial t}
\end{aligned}$$

for the HJB-PDE

$$\frac{\partial V}{\partial t} + \max_u\left\{f(x,u)\bullet\nabla V + L(x,u)\right\} = 0\,, \qquad (2.13)$$

with terminal time condition $V(t_f, x) = \Phi(x)$, Equation (2.13) is solved backward in time to determine the function *V*(*t*, *x*). The important observation, however, is that V is a means to an end rather than the desired quantity. That is, the value u^* of *u* for which the maximum is attained is the maximizing control function: if we have computed the value function V for each time and possible state value, then

$$u^*(t,x) = \arg\max_u\left\{f(x,u)\bullet\nabla V + L(x,u)\right\}. \qquad (2.14)$$

The methods we employ in this project rely directly on the Bellman equation (2.10) to compute the value function before using (2.14) to extract the controller. The algorithms use the algebraic device of max-plus arithmetic, which we define after considering stochastic problems.

Stochastic control problems generally begin with a stochastic differential equation

$$dX = f(X,u)dt + \sigma(X)dW, \quad X(t_0) = x_0, \tag{2.15}$$

in which W is a standard Brownian motion. We define the cost functional

$$J(u,x_0,t_0) = E\left[\int_{t_0}^{t_f} L(X(t),u(t))dt + \Phi(X(t_f))\right], \tag{2.16}$$

which is to be maximized over admissible controls, which are progressively measureable and satisfy

$$u \in U(t_0,t_f) \subset \left\{ u:[t_0,t_f] \to \mathbf{R}^m \mid E\int_{t_0}^{t_f} |u|^2 < \infty \right\}. \tag{2.17}$$

The Bellman equation for the value function in the stochastic case requires an expectation

$$V(t,y) = \max_{u \in U(t,s)} \left\{ E\left[\int_t^s L(X(\tau),u(\tau))d\tau + V(s,X(s;t,y,u))\right]\right\}, \tag{2.18}$$

in which *V* is the value function is once again the optimal objective:

$$V(x_0,t_0) = \max_{u \in U(t_0,t_f)} \left(J(u,x_0,t_0)\right). \tag{2.19}$$

As in the deterministic case, the value function is usually characterized through the HJB PDE, which includes a diffusion term that arises from the Brownian perturbation to the dynamics. This equation is given by

$$\frac{\partial V(t,x)}{\partial t} + \max_u \left\{ \sum_{ij} \partial_{x_i} \partial_{x_j} \left(\sigma(x,u)\sigma^T(x,u)V(t,x)\right) + f(x,u) \bullet \nabla V(t,x) + L(x,u,t) \right\} = 0, \tag{2.20}$$

in which the maximization is over $\mathbf{R}^m$. The optimal control is determined, as a function of the time and state variables, as the argument that attains the max in the HJB, or alternatively, piecewise as the argument that attains the max in the Bellman equation (2.18).

As opposed to deterministic problems, discrete stochastic control problems arise in two distinct modeling forms. The first is a difference equation model that is quite analogous to the deterministic model:

$$X(t+1) = f(X(t), u(t), \eta(t)), \quad X(t_0) = x_0, \tag{2.21}$$

in which η denotes the plant noise process. The objective functional to be maximized is

$$J(u, x_0, t_0) = E\left[\sum_{k=0}^{N-1} L(X(t_0+k), u(t_0+k)) + \Phi(X(t_f))\right], \tag{2.22}$$

with the expectation added to (2.2). The Bellman equation becomes

$$V(t,x) = \max_{u(t)} \left\{ L(x,u) + E\left[V(t+1, f(X(t), u(t)))\right]\right\}. \tag{2.23}$$

We note here for completeness, and for future use within the max-plus computational context, that a second commonly used discrete time stochastic model is that of a controlled Markov chain. Markov chains operate with a state variable *X*, living in a discrete space, *S*, usually taken to be *S* = {1,2,3,…,*N*}. The model is the transition matrix

$$P_{ij}(u) = \Pr\left[X(t+1) = j \mid X(t) = i, u\right], \tag{2.24}$$

giving dynamics in terms of transition probabilities rather than the direct functional propagation of (2.21). The cost function is still defined as in (2.22), but due to the expectation and distinct probabilistic structure the Bellman equation takes a different form:

$$V(t,x) = \max_{u(t)} \left\{ L(x,u) + \sum_{y=1}^{N} P_{xy} V(t+1, y) \right\}. \tag{2.23}$$

Having built up the background on the deterministic and stochastic structures of interest for our nonlinear control computations, we turn to the algebraic device of max-plus arithmetic.

3.0 The Max-Plus Arithmetic

The max-plus view of dynamic programming begins with some arithmetic definitions. We consider the extended real numbers augmented by $-\infty : \mathbf{R}^{-} = \mathbf{R} \cup \{-\infty\}$. On this set, we define two operations, $\oplus$ and $\otimes$, by

$$\begin{aligned} a \oplus b &= \max\{a, b\} \\ a \otimes b &= a + b \end{aligned}. \tag{3.1}$$

The inclusion of $-\infty$ is necessary to provide an additive identity or "zero" element: $a \oplus -\infty = \max\{a, -\infty\} = a$. The number 0 actually becomes the multiplicative identity or "one" element: $a \otimes 0 = a + 0 = a$. The set $\mathbf{R}^{-}$ together with these operations forms a commutative semi-ring.

Within the context of our notation, the axioms of a semi-ring are that the operations and the set obey the following. For all *a, b, c*, we have

- $a \oplus b = b \oplus a$
- $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
- $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$.

Moreover, we require that

- a unique additive identity exists, as does a unique multiplicative identity *e*
- the additive identity annihilates **R** under multiplication: $\theta \otimes a = \theta$.

These properties are straightforward to establish in the max-plus arithmetic. A difficulty arises because addition is idempotent. Since $a \oplus a = \max\{a, a\} = a$, the semi-ring cannot be extended to a ring (or field): the only idempotent ring is the zero ring. Thus, we are unable to extend the set to obtain a max-plus version of subtraction. This issue prevents us from using the well-developed functional analytic machinery of linear operators and operator semigroups on Banach or Hilbert spaces, as the scalar set is not a field. As we shall see, the max-plus structure offers some interesting computational opportunities, but there are prices to be paid in its use.

From the scalar addition and multiplication operations, we build standard linear algebraic objects, such as matrices and vectors. If we consider an $n \times n$ array, A, of elements of $\mathbf{R}^-$ and a column vector, x, of n elements of $\mathbf{R}^-$, define the max-plus matrix-vector product $y = A \otimes x$ by

$$y_i = \bigoplus_{j=1}^{n} \left(A_{i,j} \otimes x_j \right) = \max_j \left\{ A_{i,j} + x_j \right\}. \tag{3.2}$$

Similarly, we may define max-plus matrix multiplication and addition:

$$\left(A \otimes B \right)_{i,j} = \max_k \left\{ A_{i,k} + B_{j,k} \right\}, \qquad \left(A \oplus B \right)_{i,j} = \max \left\{ A_{i,j}, B_{i,j} \right\}. \tag{3.3}$$

Raising a matrix to a power, then, is repeated applications of multiplication. Eigenvalue problems in the max-plus setting are given by

$$\begin{aligned} & A \otimes x = \lambda \otimes x \\ & \qquad \Leftrightarrow (A \otimes x)_i = \left(\lambda \otimes x \right)_i \\ & \qquad \Leftrightarrow \max_k \left\{ A_{i,k} + x_k \right\} = \lambda + x_i \end{aligned} \quad . \tag{3.4}$$

Function spaces can be constructed as well. In particular, we use the standard sets of functions on domains of interest for the dynamical systems we control. We equip these sets with max-plus arithmetic in the range space:

$$\left(f \oplus g \right)(x) = \max\{ f(x), g(x) \}, \quad \left(f \otimes g \right)(x) = f(x) + g(x) . \tag{3.5}$$

The max-plus analytic and functional analytic concepts are also relevant. For example, we may be interested in infinite sums and even integrals, which require replacing the max operation with supremum. We abuse the max notation by assuming the convention that max means sup when sup is needed. For any set S, finite or infinite, we use the notation

$$\max_{s\in S} f(s) = \sup_{s\in S} f(s) \ . \tag{3.6}$$

For bounded continuous functions on a compact domain, the max of course exists, while for bounded functions generally we need the supremum. Max-plus Riemann sums take the form

$$\bigoplus_{i=1}^{n} f(s_i)\otimes \Delta x_i = \max_{1\le i\le n}\{ f(s_i)+\Delta x_i\}, \tag{3.7}$$

leading to the following definition of Riemann (and Lebesgue) integral:

$$\int_S^{\oplus} f = \max_{s\in S} f(s)\,, \tag{3.8}$$

as the term vanishes in the defining limit. More generally, we may define a measure μ on (a σ -algebra of sets in) S by the usual axioms, appropriately translated into max-plus arithmetic:

$$\begin{aligned} &(i) \quad \mu(\varnothing) = -\infty \\ &(ii) \quad \mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sup_i \mu(A_i), \text{ when } \{A_i\} \text{ are pairwise disjoint.} \end{aligned} \tag{3.9}$$

For a max-plus μ measure to have a density g, we have $\mu(A) = \sup_{x\in A} g(x)$, and in this case we define the integral by

$$\int_S^{\oplus} f d\mu = \max_{s\in S}\{f(s)+g(s)\}. \tag{3.10}$$

Another key issue in max-plus functional analysis is the approximation of functions. Finite difference and finite element methods are often used in the approximation of solutions of differential equations. Finite elements will play a crucial role in our computational schemes, so we introduce the max-plus finite element expansion to approximate a function *f*:

$$f^N(x) = \bigoplus_{k=1}^{N} a_k \otimes \psi_k(x) = \max_k \{a_k + \psi_k(x)\} \tag{3.11}$$

in which the basis functions ψ_k are functions chosen for computational simplicity and approximation efficiency. The coefficients a_k are determined using max-plus analogs of standard finite element projections:

$$a_i = -\max_x \{\psi_i(x) - f(x)\}. \tag{3.12}$$

This projection differs from the standard projection primarily due to the idempotent nature of the max-plus addition operation. Before discussing the nature of this computation, we consider some finite element basis families. In particular, we introduce three families of finite element types: linear elements

$$\psi_i(x) = -c_i |x - x_i|, \tag{3.13}$$

quadratic elements

$$\psi_i(x) = -c_i |x - x_i|^2, \tag{3.14}$$

and Legendre elements

$$\psi_i(x) = p_i^T x. \tag{3.15}$$

In the linear and quadratic elements, x_i are the element nodes, and c_i are scale parameters. These elements are completely analogous to traditional finite element linear splines. In the Legendre formulation, the elements are defined by the slopes p_i. The Legendre elements are particularly interesting in that the Legendre transform

$$\hat{f}(p) = \max_x \{f(x) - p^T x\}, \tag{3.16}$$

is the max-plus analog of the Fourier transform for convex functions. The function f is reconstructed from its Legendre transform through the inverse transform

$$f(x) = \max_p \left\{ \hat{f}(p) - p^T x \right\}, \tag{3.17}$$

whose action is identical to the forward transform, leading to a max-plus spectral approximation type.

Max-plus finite elements lead to a lower envelope type of approximation. Figure (3.1) illustrates the linear and quadratic approximations of a simple continuous function.

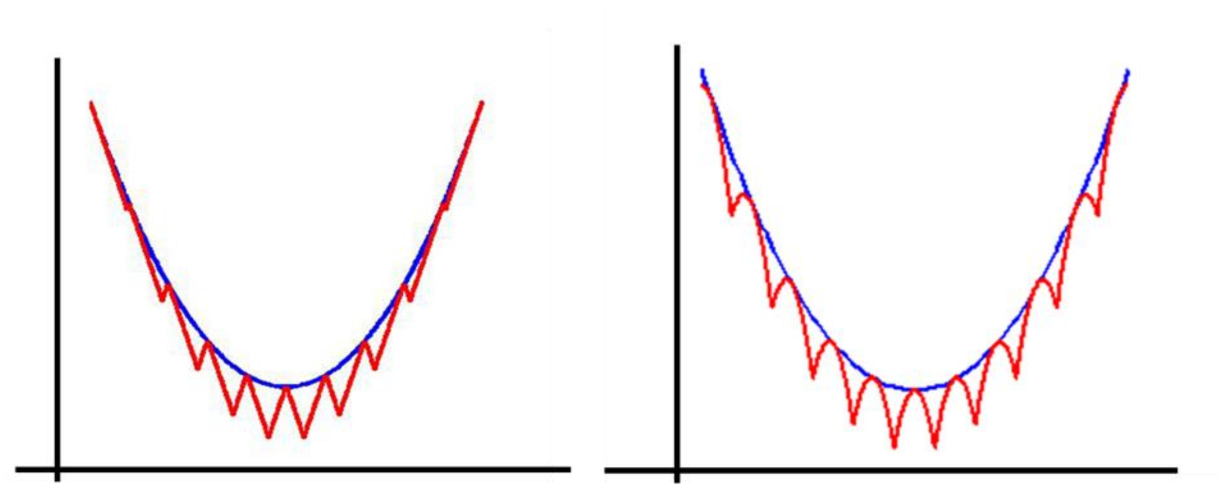


Figure 3.1 Linear and Quadratic Max-Plus Finite Element Approximations

The Legendre elements provide a lower envelope with supporting straight lines. The process and resulting approximation are shown in Figure 3.2.

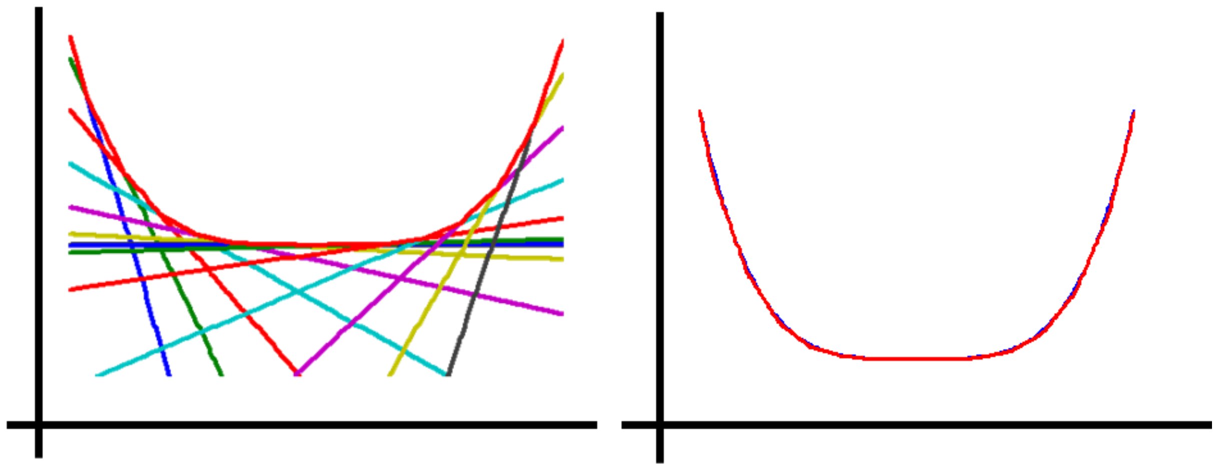


Figure 3.2 The supporting lines and the resulting approximation of max-plus Legendre elements

Details of the max-plus arithmetic, its properties and applications, can be found in [BCOQ, CGQ, HOW, KM, L, LMS1, LMS2, Mc2]. Our interest in this seemingly abstract concept is its applicability for nonlinear control computation. With the basic arithmetic and approximational tools in place, we return to the deterministic control problem.

4.0 Section Nonlinear Control and The Max-Plus Algebra

We begin with an examination of the discrete deterministic control problem (2.1) and (2.2), whose Bellman equation (2.4) we recall as

$$V(t,x) = \max_{u(t)} \{L(x,u) + V(t+1, f(x(t),u(t)))\}, \tag{4.1}$$

The applicability of and interest in the max-plus algebra arise from the fact that (4.1) is a max-plus linear equation. To establish this fact, we consider the operator

$$A\phi(y) = \max_{u} \{L(y,u) + \phi(f(y,u))\}. \tag{4.2}$$

We first note that

$$\begin{aligned} A(a \otimes \phi)(y) &= \max_{u} \{L(y,u) + a + \phi(f(y,u))\} \\ &= a + \max_{u} \{L(y,u) + \phi(f(y,u))\} \\ &= a \otimes A\phi(y), \end{aligned}$$

so that scalar multiples pass through the operation. Next, we see that

$$\begin{aligned} A(\psi \oplus \phi)(y) &= \max_{u} \{L(y,u) + \max\{\psi(f(y,u)), \phi(f(y,u))\}\} \\ &= \max_{u} \{\max\{L(y,u) + \psi(f(y,u)), L(y,u) + \phi(f(y,u))\}\} \\ &= \max\left\{\max_{u} \{L(y,u) + \psi(f(y,u))\}, \max_{u} \{L(y,u) + \phi(f(y,u))\}\right\} \\ &= (A\psi \oplus A\phi)(y), \end{aligned}$$

so that *A* is max-plus linear. The Bellman equation becomes a discrete time linear dynamical system in the max-plus sense, with a max-plus operator exponential solution:

$$V(t,x) = A^{\otimes\, t_f - t} V(t_f, x)\,. \tag{4.3}$$

Likewise, the continuous time Bellman equation leads us to construct a family of linear operators

$$S_{t,s}(\phi)(y) = \max\left\{ \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \phi(x(s;t,y,u)) \right\}, \tag{4.4}$$

that propagate backward in time from *s* to *t*: the Bellman equation becomes $S_{t,s}(V(s,\bullet))(y) = V(t,y)$. Further, we note that, since *f* and *L* do not depend explicitly on time, *S* only depends on the time difference *s* – *t*. We thus adjust our notation, using S_{s-t} in place of $S_{t,s}$. Computing the value function can be viewed as iteratively evaluating this operator:

$$V(t-h,y) = S_h(V(t,\bullet))(y). \tag{4.5}$$

We establish that S_{s-t} is a max-plus linear operator in a manner similar to the discrete time case:

$$\begin{aligned}
S_{s-t}(\phi \oplus \psi)(y) &= \max_u \left\{ \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \max\{\phi(x(s;t,y,u)), \psi(x(s;t,y,u))\} \right\} \\
&= \max_u \left\{ \max\left\{ \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \phi(x(s;t,y,u), \right.\right. \\
&\qquad \left.\left. \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \psi(x(s;t,y,u)) \right\}\right\} \\
&= \max\left\{ \max_u \left\{ \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \phi(x(s;t,y,u) \right\}, \right. \\
&\qquad \left. \max_u \left\{ \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \psi(x(s;t,y,u)) \right\}\right\} \\
&= S_{s-t}(\phi)(y) \oplus S_{s-t}(\psi)(y)
\end{aligned}$$

So that additivity is thus established. Next, we examine

$$\begin{aligned}S_{s-t}(a\otimes\phi)(y) &= \max_u\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + a + \phi(x(s;t,y,u))\right\}\\ &= a + \max_u\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \phi(x(s;t,y,u))\right\}\\ &= a\otimes S_{s-t}(\phi)(y)\end{aligned}$$

so that scalar multiplicative factors commute with the operator. Thus S_{s-t} is a max-plus linear operator, and the Bellman equation's solution can be written as an operator exponential solution:

$$V(t_f - nh, x) = S_h^{\otimes n} V(t_f, x)\,. \tag{4.6}$$

One issue is of course the selection of the time step h. Another is that the continuous state space requires a discretization as well. Toward that end, we introduce a finite element expansion of the value function:

$$V^N(t,x) = \bigoplus_{k=1}^{N} a_k(t)\otimes\psi_k(x) = \max_k\{a_k(t)+\psi_k(x)\}. \tag{4.6}$$

Inserting the expansion into the Bellman operator, we have

$$\begin{aligned}S_{s-t}(\bigoplus_{i=1}^{N}(a_i\otimes\psi_i)(y) &= \max_u\left\{\int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \max_{1\le i\le N}\{a_i+\psi_i(x(s;t,y,u))\}\right\}\\ &= \max_{1\le i\le N}\left\{a_i + \max_u \int_t^s L(x(\tau;t,y,u),u(\tau))d\tau + \psi_i(x(s;t,y,u))\right\}\\ &= \bigoplus_{i=1}^{N} a_i\otimes S_{s-t}(\psi_i)(y)\end{aligned}$$

so that the Bellman operator's application to the basis becomes one key component. A second is the projection of the result back onto the basis:

$$\bigoplus_{i=1}^{N} a_i\otimes S_{s-t}(\psi_i)(y) \approx \bigoplus_{i=1}^{N} b_i\otimes\psi_i(y)\,, \tag{4.7}$$

or more precisely

$$V^N(t,x) = \overset{N}{\underset{k=1}{\oplus}} a_k(s) \otimes S_{s-t}(\psi_k)(x) \approx \overset{N}{\underset{k=1}{\oplus}} a_k(t) \otimes \psi_k(x)\,. \tag{4.8}$$

That is, just as in traditional finite elements, we seek a means of propagating the finite element expansion coefficients directly. There are two primary approaches to this propagation. The first, and simplest, is recommended by McEneaney [Mc2]:

$$a(t) = B \otimes a(s)\,, \tag{4.9}$$

in which the matrix B is defined by

$$B_{ij} = -\max_x \left\{ \psi_i(x) - S_{s-t}(\psi_j)(x) \right\}. \tag{4.10}$$

Note that this form applies (3.12) to project $S_{s-t}(\psi_j)$ onto the basis.

An alternative uses a max-plus variant of mixed finite elements. That is, the variational formulation

$$\max_y \{V(t,y) + \phi(y)\} = \max_y \{S_{t,s}(V(s,\bullet)(y) + \phi(y)\} \tag{4.11}$$

is solved using distinct basis and test functions. We let $\phi_1, \cdots, \phi_M$ denote the finite set of test functions, with $\psi_1, \cdots, \psi_N$ denoting the basis as above. We expand the value functions at the two time steps in terms of the basis:

$$\begin{aligned} V^N(t,x) &= \overset{N}{\underset{k=1}{\oplus}} a_k(t) \otimes \psi_k(x) = \max_k \left\{ a_k(t) + \psi_k(x) \right\} \\ V^N(s,x) &= \overset{N}{\underset{k=1}{\oplus}} a_k(s) \otimes \psi_k(x) = \max_k \left\{ a_k(s) + \psi_k(x) \right\} \end{aligned}\,, \tag{4.12}$$

and we insert these expansions into the variational form (4.11):

$$\max_k \left\{ a_k(t) + \max_y \{\psi_k + \phi_i(y)\} \right\} = \max_k \left\{ a_k(s) + \max_y \{S_{t,s}(\psi_k)(y) + \phi_i(y)\} \right\}, \tag{4.13}$$

which is a max-plus analog of the traditional finite element form $Ma(t) = Ka(s)$ with mass matrix and stiffness matrix K. In general, the max-plus mass matrix cannot be inverted, but a maximal subsolution exists:

$$\left(M^{\#}b\right)_j = \min_i \{-M_{ij} + b_i\} \tag{4.14}$$

leading to a max-plus iteration of the form

$$a_j(t) = \min_i \left\{ -\max_y \{\phi_j(y) + \psi_i(y)\} + \max_k \left\{ a_k(s) + \max_y \{S_{t,s}(\psi_k)(y) + \phi_i(y)\} \right\} \right\} \tag{4.15}$$

a slightly more involved but more efficient (in terms of basis size) computation. The test-basis estimation produces an upper/lower envelope of the approximated function, as illustrated in Figure 4.1.

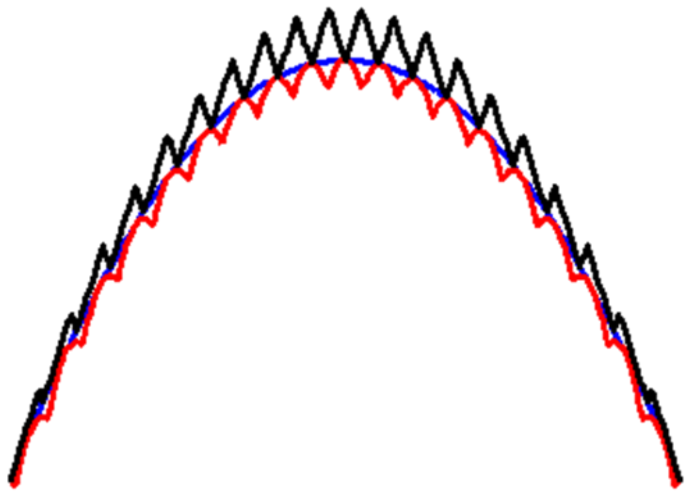


Figure 4.1 Mixed finite element test and basis functions trapping a function to be approximated

Two final issues to be addressed prior to computational implementation: the computation of the Bellman operator and the determination of the optimal control. The Bellman operator, we recall, is

$$S_h(\phi)(y) = \max \left\{ \int_0^h L(x(\tau;t,y,u),u(\tau))d\tau + \phi(x(h;0,y,u)) \right\}, \tag{4.16}$$

which requires maximization over control functions on the interval [0, *h*] as well as solution of the differential equation $\dot{x} = f(x,u)$. Analytic expressions will not be available in general, so we need to approximate the optimization and integration.

A first, simplest approach is to assume h is sufficiently small that we may approximate the control as constant, the running cost integrand as constant, and the dynamics with one step of an Euler integration. This leads to the approximation

$$S_h^0(\phi)(y) = \max_u \{hL(y,u) + \phi(y + hf(y,u))\}. \tag{4.17}$$

A slightly better approximation involves a trapezoid approximation to the integral:

$$S_h^1(\phi)(y) = \max_u \left\{ \frac{h}{2}\big(L(y,u) + L(y+hf(y,u),u)\big) + \phi(y + hf(y,u)) \right\}. \tag{4.18}$$

Both of these operator approximations involve maximization over a scalar control value, which is applied as a constant over a short time. A further refinement involves approximating with a control that changes linearly over the short time interval. The operator approximation is

$$S_h^1(\phi)(y) = \max_{u,v} \left\{ \frac{h}{2}\big(L(y,u) + L(y+hf(y,u),v)\big) + \phi(y + hf(y,u)) \right\}. \tag{4.19}$$

Finer approximations still can be had by dividing the interval [0, *h*] into subintervals.

More involved approximations divide the time interval [0, *h*] into subintervals. We may use piecewise linear controls, higher order integration techniques on the running cost, and higher order approximations to the differential equation solution.

To compute the optimal control from the value function, we have found two methods that work efficiently and effectively. The first uses the maximizing control value from the Bellman operator computation:

$$u^*(y,t) = \arg\max_u \left\{ \frac{h}{2}\big(L(y,u) + L(y+hf(y,u),u)\big) + V^N(t+h, y + hf(y,u)) \right\}. \tag{4.20}$$

The second plugs the value function into the Hamilton-Jacobi equation:

$$u^*(y,t) = \arg\max_u \left\{ L(y,u) + \nabla V^N \bullet f(y,u) \right\}. \tag{4.21}$$

Either (4.20) or (4.21) can be used to extract the controller from the value function. We note, however, that the "scalloping" of the max-plus approximate value function, as illustrated in Figures 3.1 and 4.1, can have a strong impact when differentiated in (4.20). Some amount of spatial filtering may be applied to smooth the value prior to computing its gradient: that is, we use (4.21) in conjunction with a mollifying convolution filter:

$$V^N(t,x) = \int_D V^N(t,y) W(t,x-y) dy . \tag{4.22}$$

5.0 A Max-Plus Approach to Stochastic Control

Recall the controlled stochastic differential equation

$$dX = f(X,u)dt + \sigma(X)dW, \quad X(t_0) = x_0 , \tag{5.1}$$

with objective functional

$$J(u,x_0,t_0) = E\left[\int_{t_0}^{t_f} L(X(t),u(t))dt + \Phi(X(t_f)) \right], \tag{5.2}$$

which is to be maximized over admissible controls. The dynamic programming propagation operator in the stochastic case takes the form

$$S_{s,t}(\phi) = \max\left\{ E\left[\int_t^s g(X(\tau),u(\tau)d\tau + \phi(X(s;t,y,u)) \right] \right\}, \tag{5.3}$$

and the difficulty is that this operator is not max-plus linear due to the expectation. While scalar multiplication (addition in traditional arithmetic) factors through the expectation:

$$\begin{aligned}\int_\Omega (a\otimes\phi(\omega))dP(\omega) &= \int_\Omega (a+\phi(\omega))dP(\omega)\\ &= \int_\Omega a\,dP(\omega)+\int_\Omega \phi(\omega)dP(\omega)\\ &= a+\int_\Omega \phi(\omega)dP(\omega) = a\otimes\int_\Omega \phi(\omega)dP(\omega),\end{aligned}$$

max-plus addition (maximization) is not respected by expectation:

$$\begin{aligned}\int_\Omega (\psi(\omega)\oplus\phi(\omega))dP(\omega) &= \int_\Omega \max\{\psi(\omega),\phi(\omega)\}dP(\omega)\\ &\neq \max\left\{\int_\Omega \psi(\omega)dP(\omega),\int_\Omega \phi(\omega)dP(\omega)\right\}\\ &= \int_\Omega \psi(\omega)dP(\omega)\oplus\int_\Omega \phi(\omega)dP(\omega).\end{aligned}$$

The lack of linearity prohibits direct application of the max-plus exponential solution operator. We may treat stochastic problems as nonlinear, or we may reconsider our definition of stochasticity.

Considering stochastic problems as max-plus nonlinear, we may find some computational efficiencies through the use of distributivity. Defining $I=\{1,2\}, J_I=\{(1,1),(1,2),(2,1),(2,2)\},$ we note that

$$\begin{aligned}&(a_{1,1}\oplus a_{1,2})\otimes(a_{2,1}\oplus a_{2,2}) = \bigotimes_{i=1}^{2}\bigoplus_{j=1}^{2}(a_{i,j})\\ &(a_{1,1}\otimes a_{2,1})\oplus(a_{1,1}\otimes a_{2,2})\oplus(a_{1,2}\otimes a_{2,1})\oplus(a_{1,2}\otimes a_{2,2}).\\ &= \bigoplus_{(j_1,j_2)\in J_I}\bigotimes_{i=1}^{2}(a_{i,j_i})\end{aligned}$$

More generally, when N = {1,2,…,*n*}, M = {1,2,…,*n*}, and J_M = the set of all ordered *n*-tuples of elements of *M*, then

$$\bigotimes_{i=1}^{n}\bigoplus_{j=1}^{m}(a_{i,j}) = \bigoplus_{(j_1,\cdots,j_n)\in J_M}\bigotimes_{i=1}^{n}(a_{i,j_i}). \tag{5.4}$$

Note that the number of summands has increased dramatically on the right side of the equality. This distributive property is generalized further in the following result.

Theorem 1. Suppose that *W* and *Z* are separable metric spaces. Suppose that *P* is a finite Borel measure on *W*. Suppose that h is a measureable function on *W* x *Z* satisfying the following condition: for every $\varepsilon > 0$ and every $w \in W$, there exists a $\delta > 0$ such that

$$|h(w,z) - h(w',z)| < \varepsilon, \quad \forall z \in Z \text{ and } w' \in B_\delta(w).$$

Then,

$$\int_W \sup_{z \in Z} h(w,z)\, dP(w) = \sup_{f \in M(W,Z)} \int_W h(w, f(w))\, dP(w), \tag{5.5}$$

in which *M*(*W*, *Z*) denotes the set of Borel measureable functions mapping *W* to *Z*.

To apply this result, we develop max-plus finite element approximations to the Bellman equations.

$$\begin{aligned} S_{s,t}(V^N) &= \max\left\{ E\left[\int_t^s L(X(\tau), u(\tau) d\tau \otimes \left(\bigoplus_{i=1}^N a_i(s) \otimes \psi_i(X(s;t,y,u)) \right) \right] \right\} \\ &= \bigoplus_u \left\{ \int_\Omega \left[\bigoplus_{i=1}^N \left(\int_t^s L(X(\tau), u(\tau) d\tau \otimes a_i(s) \otimes \psi_i(X(s;t,y,u)) \right) \right] dP \right\} \end{aligned}. \tag{5.6}$$

Applying Theorem 1, we have

$$S_{s,t}(V^N)(y) = \bigoplus_{u \in U(s,t)} \bigoplus_{Z \in M(\Omega, I_N)} \left\{ \int_\Omega \int_s^t L(X_\tau, u_\tau) d\tau \otimes \int_\Omega a_{Z(\omega)}(t) \otimes \psi_{Z(\omega)}\big(X(t;s,y,u)\big) dP(\omega) \right\}, \tag{5.7}$$

in which the distributivity property of the theorem involves the set of random variables taking values in the set {1,2, …, *N*} for the interchange of expectation and maximization order. This propagation is then inserted into the variational form

$$\max_y \{V(t,y) + \phi(y)\} = \max_y \{S_{t,s}(V(s,\bullet)(y) + \phi(y)\}. \tag{5.8}$$

Expanding the one-time-step-back value function in terms of the basis leads to

$$a_j(t) = \min_i \left\{ -\max_y \{\phi_j(y) + \psi_i(y)\} + \max_y \{S_{t,s}(V(s,\bullet))(y) + \phi_i(y)\} \right\}, \tag{5.9}$$

providing for the temporal propagation of the coefficients of the finite element expansion. This time-stepper remains a nonlinear operation, as we cannot move the basis expansion of the value function through the propagator *S*.

An alternate, more efficient approach to stochasticity uses a max-plus definition of probability. A max-plus probability space is a triplet (Ω,**F**,*Q*) of a set Ω called the sample space, a σ-field **F** of subsets of Ω, and a max-plus probability measure *Q* that satisfies the following:

$$\begin{aligned}
&Q(\Omega) = 0, \\
&Q(\varnothing) = -\infty, \\
&\{A_k\}_{k=1}^{\infty} \subset \mathbf{F}, \text{mutually exclusive} \\
&\qquad \Rightarrow \bigoplus_{k=1}^{\infty} Q(A_k) = \max_k \{Q(A_k)\} = Q\left(\bigcup_{k=1}^{\infty} A_k\right).
\end{aligned}$$

These criteria are completely analogous to the standard definition of probability measures in “plus-times” arithmetic. The inclusion of the constraint Q(Ω) = 0 makes the measures of (3.9) into probability measures. Note that max-plus probabilities have some unintuitive properties. For example,

$$\max\{Q(A^c), Q(A)\} = 0.$$

For the probability measure *Q* to have a density *q*, we must have

$$Q(A) = \max_{x \in A} \{q(x)\}.$$

Max-plus probabilities can be derived from standard probability measures through a large deviations approach. Max-plus arithmetic can be viewed as a limit of log-plus arithmetic:

$$\begin{aligned}
a \oplus b &= \lim_{\varepsilon \to 0} a \oplus_{\varepsilon} b \ \text{ where } \ a \oplus_{\varepsilon} b = \varepsilon \log\bigl(\exp(a/\varepsilon) + \exp(b/\varepsilon)\bigr), \\
a \otimes b &= \lim_{\varepsilon \to 0} a \otimes_{\varepsilon} b \ \text{ where } \ a \otimes_{\varepsilon} b = \varepsilon \log\bigl(\exp(a/\varepsilon)\exp(b/\varepsilon)\bigr).
\end{aligned}$$

Log-plus operations enter into probabilistic modeling in the following way. We consider a family of probability measures $(P_\varepsilon)_{\varepsilon>0}$ on the measureable space $(\Omega,\mathbf{F})$. If this family obeys a large deviation principle, then the limit

$$Q(A) = \lim_{\varepsilon\to 0} \varepsilon \log P_\varepsilon(A), \tag{5.10}$$

exists and is a max-plus probability measure. As a simple example, consider a family of Gaussian densities

$$p_\varepsilon(x) = \frac{1}{\sqrt{2\pi\varepsilon}} \exp\left(-\frac{1}{2\varepsilon}|x-\mu|^2\right), \tag{5.11}$$

with vanishing variance $\varepsilon \rightarrow 0$. Then the measure Q is characterized by the quadratic density

$$q(x) = -\frac{1}{2}|x-\mu|^2 \ . \tag{5.12}$$

This process gives us a method of constructing max-plus measures from exponential density families.

Expectations in max-plus probability are defined by the following:

$$E^\oplus(X) = \max_\omega\{X(\omega) + q(\omega)\}, \tag{5.13}$$

with which we can define objective functions for max-plus stochastic control problems.

One challenge here is that max-plus probability requires a minimizing structure for the control problem. That is, we redefine the running cost and terminal costs to be their negatives:

$$\tilde{L}(x,u) = -L(x,u), \quad \tilde{\Phi}(x) = -\Phi(x), \tag{5.14}$$

for a control problem of minimizing

$$\begin{aligned} J(u,x_0,t_0) &= E^{\oplus}\left[\int_{t_0}^{t_f} \tilde{L}(X(t),u(t))dt + \tilde{\Phi}(X(t_f))\right] \\ &= \max_{\omega}\left[\int_{t_0}^{t_f} \tilde{L}(X(t),u(t))dt \otimes \tilde{\Phi}(X(t_f)) \otimes p(\omega)\right] \end{aligned}, \qquad (5.15)$$

so that the value function becomes

$$\begin{aligned} V(t,x) &= \min_u E^{\oplus}\left[\int_{t}^{t_f} \tilde{L}(X(t),u(t))dt + \tilde{\Phi}(X(t_f))\right] \\ &= \min_u \max_{\omega}\left[\int_{t_0}^{t_f} \tilde{L}(X(t),u(t))dt \otimes \tilde{\Phi}(X(t_f)) \otimes p(\omega)\right] \end{aligned}. \qquad (5.16)$$

This formulation leads to a game, in which the control designer is playing against the stochastic player.

The relationship between max-plus expectation (and hence objective functionals) and traditional stochastic expectation can be examined within the framework of large deviations and risk sensitive control, as (5.10) leads to

$$\lim_{\varepsilon \to 0} \varepsilon \log E\left[e^{J/\varepsilon}\right] = E^{\oplus}[J]. \qquad (5.17)$$

The max-plus expectation is thus a risk sensitive limiting objective functional.

The value function for the max-plus expected objective minimization is given by

$$\begin{aligned} V(t,x) &= \min_u E^{\oplus}\left[\int_{t}^{s} \tilde{L}(X(t),u(t))dt \otimes V(s,X(s))\right] \\ &= \min_u \max_{\omega}\left[\int_{t}^{s} \tilde{L}(X(t),u(t))dt \otimes V(s,X(s)) \otimes p(\omega)\right] \end{aligned} \qquad (5.18)$$

Expanding V in terms of a max-plus basis leads to

$$
\begin{aligned}
V(t,x) &\approx \min_u E^{\oplus}\left[\int_t^s \tilde{L}(X(t),u(t))dt \otimes \left(\bigoplus_{i=1}^{N} a_i(s)\otimes\psi_i(X(s))\right)\right] \\
&= \min_u \bigoplus_{i=1}^{n} a_i(s)\otimes E^{\oplus}\left[\int_t^s \tilde{L}(X(t),u(t))dt\otimes\psi_i(X(s))\right] \\
&= \min_u \max_i \left\{a_i(s)+\max_{\omega}\left[\int_t^s \tilde{L}(X(t),u(t))dt+\psi_i(X(s))+p(\omega)\right]\right\}
\end{aligned}
\quad . \tag{5.19}
$$

Here we define

$$
T^u_{t,s}\phi(y) = E^{\oplus}\left[\int_t^s \tilde{L}(X(t),u(t))dt\otimes\varphi(X(s))\right], \tag{5.20}
$$

for $X(t) = y$, as the short-time expected cost propagation. Then we have

$$
V(t,x)=\min_u T^u_{t,s}V(s,\bullet)(x)\approx\min_u \bigoplus_{i=1}^{n} a_i(s)\otimes T^u_{t,s}\psi_i(x)=\min_u\max_i\left\{a_i(s)+T^u_{t,s}\psi_i(x)\right\}. \tag{5.21}
$$

6.0 Max-Plus Algebra and Hardware/Software Considerations

One of the more intriguing aspects of the max-plus, min-plus, and min-max arithmetic structures is the potential to instantiate operations in hardware. Modern CPUs are designed for the efficient (vectorized) computation of multiplication and division, and we seek to leverage the concept for the efficient computation of max-plus operations.

Toward that end, we have developed, coded, and tested prototype Field Programmable Gate Array (FPGA) implementations of max-plus matrix-vector multiplication (Max-Plus MVM), which we refer to as our Max-Plus MVM FPGA. Max-Plus MVM is a mathematical computation of max-plus matrix-vector multiplication of the form:

$$
a_0 = B^n \otimes a_T \tag{6.1}
$$

in which the coefficients of the max-plus finite element expansion, or the discrete values of a value function, are iterated temporally according to the appropriately discretized Bellman equation. More specifically for a user provided matrix *B* and vector a_T Max-Plus MVM FPGA carries out Max-Plus MVM iteration *n* times on the FPGA board till either a0 is converged or the number of iterations n reaches to a predetermined number *N*.

Tempest's Max-Plus MVM FPGA system is composed of a Xilinx ML605 board (the Board), onboard circuit design and interface software suite. The Board features a Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA, and it is connected to a PC via an Ethernet 1000BASE-T cable. Interface with the Board is through Matlab. The software package is composed of set of Matlab and C codes. The package is developed and tested on a host computer with the Linux-based Ubuntu operating system. Much of the discussion of prior sections focused on the mathematical aspects of the host computer tasks, all of which boil down to two primary jobs: construction of the max-plus iteration matrices, and conversion of the value function into a controller. These tasks are essentially designing the input for and processing the output of the coprocessing subsystem. The prototype design is summarized in following diagram.

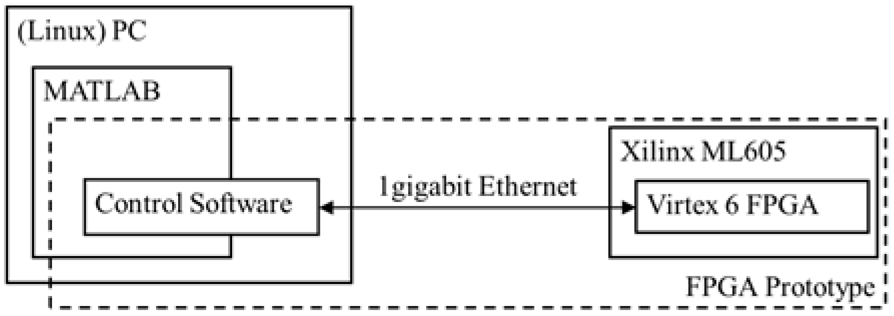


Figure 6.1 Concept block diagram Tempest's Max-Plus MVM FPGA

6.1 The Xilinx FPGA ML605.

FPGAs are two dimensional arrays of configurable logic blocks. Programmable interconnects and block configurability provide an extremely flexible computing platform. The array structure offers natural parallelism. Of course the difficulty with this level of flexibility is in the programming, which must take into account data flow and timing through the array.

The Xilinx ML605 board, seen in Figure 6.2, is a cost effective host of the FPGA device that met our computational and interface needs.

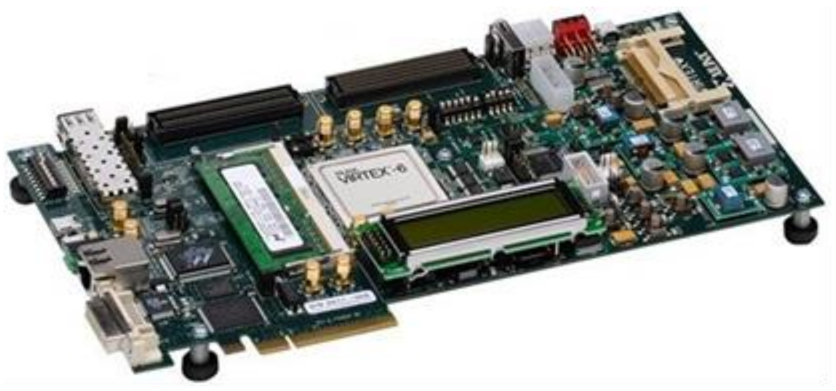


Figure 6.2 Xilinx ML605 FPGA board

The board comes with gigabit Ethernet, DDR3 memory controller and interface, DVI, system monitor, serial transceiver integration, PCI express Gen 1 and 2, and more. Xilinx provides Xilinx Intellectual Property (Xilinx-IP) Cores that are functions highly optimized for Xilinx FPGAs.

There are two communication choices provided by the Board, PCI Express and gigabit Ethernet connections. The PCIe delivers up to 16 gigabit speed and requires a PC with a PCIe slot. To accelerate development, testing, and validation, we employed an existing 1 gigabit Ethernet communication channel and driver for the board to talk to the host PC.

Xilinx provides memory interfacing Xilinx-IP core to access the DDR3 SDRAM, MT4JSF6464HY-1G1B1 on the Board. The prototype utilizes DDR3 for large size computations.

The Board contains a physical Ethernet chip, 88E1111 RCJ1 that is accessed with Xilinx-IP core for Ethernet.   It has specialized peripherals including an LCD, LEDs, dip switches and push buttons. We use only one push button for resetting the FPGA board in Max-Plus MVM FPGA system.

We utilize the built-in 32MB BPI flash as storage for the Tempest Max-Plus MVM FPGA circuit design. When the Board is initializing, the Max-Plus MVM design will be loaded automatically to the FPGA.

The details of the ML605 are provided in Figure 6.3.

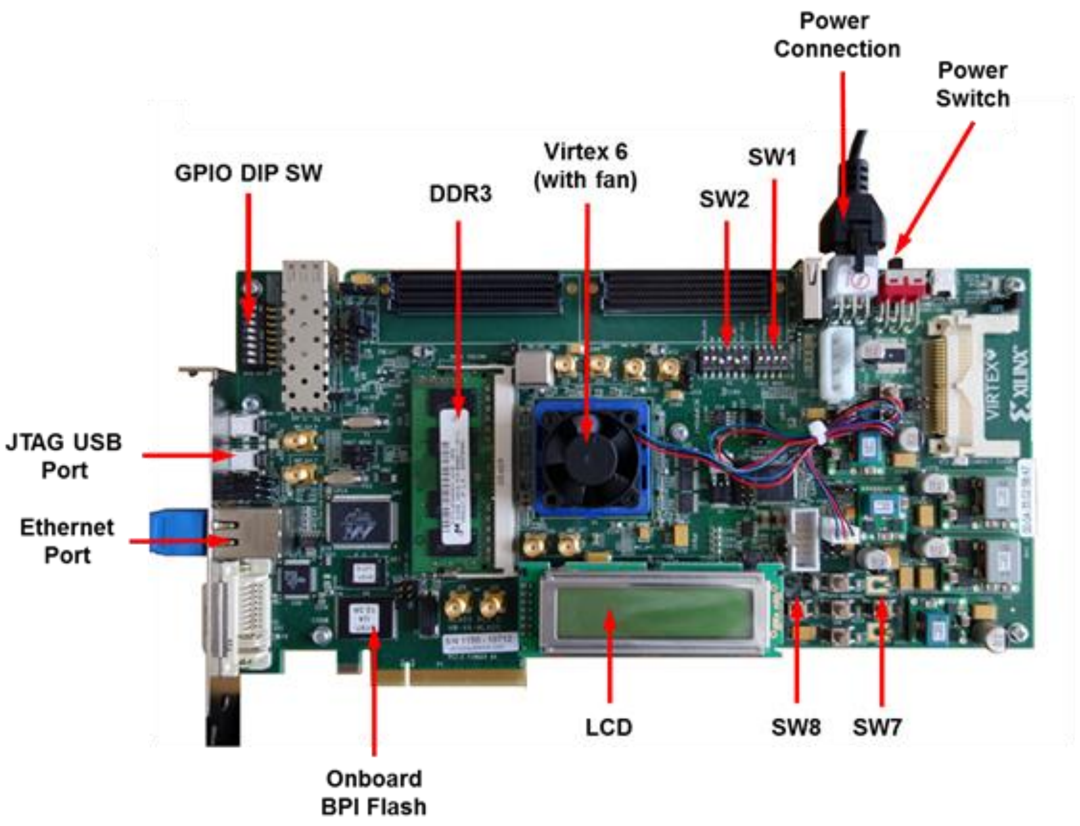


Figure 6.3 Xilinx ML605 FPGA components

The actual FPGA is the Virtex-6 chip. In this chip resides a column-oriented array of 6-input lookup table units (LUTs), multiply-and-add (MUX) units, sometimes referred to as DSPs in the FPGA literature, as well as local RAM and communication circuits. The LUTs provide the programming flexibility to instantiate all types of basic logic operations. The fine structure of the Virtex 6 array is illustrated in Figure 6.4.

Design of FPGA code involves literal flow of data through the circuit as operations are applied by the LUT and MUX units. Synchronizing the data flow through memory and communication management is a key issue in FPGA implementation.

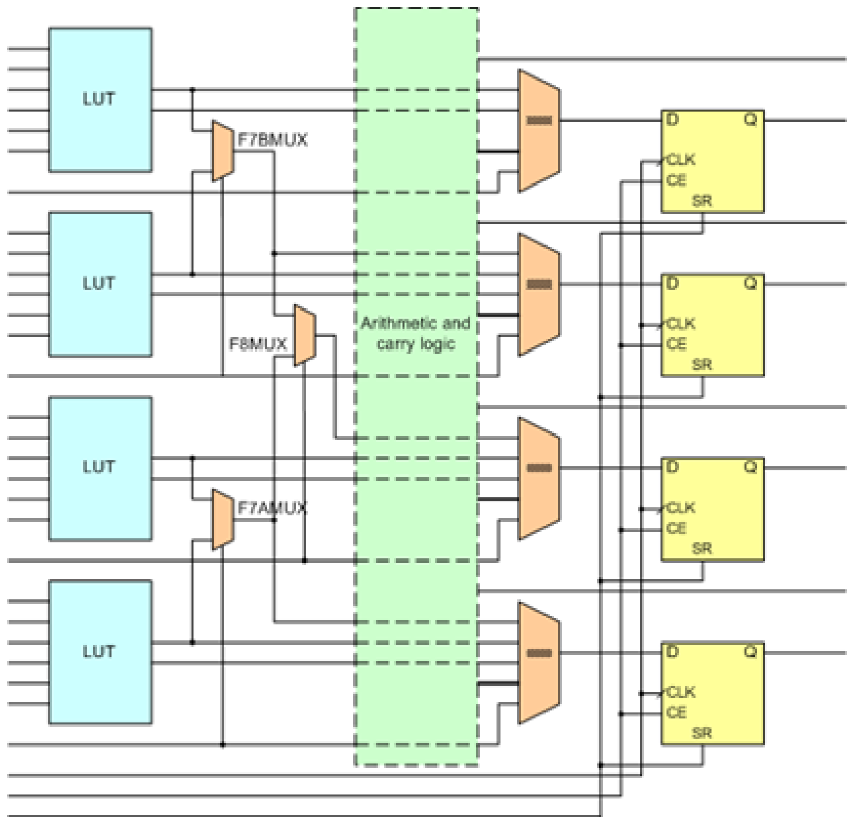


Figure 6.4 Look-up table (LUT) cells, multiply-add (MUX) cells, and fabric structure

6.2 The max-plus concept in FPGA

The base algorithm for solving dynamic programming problems is the max-plus iteration

$$\begin{aligned} a' &= B \otimes a, \\ a_i' &= \max_j \{B_{ij} + a_j\}, \end{aligned} \tag{6.2}$$

which is a parallel set of maximizations. This process must be iterated a number of times to derive the value function and controller. Each row of the matrix B can be simultaneously applied to the vector a through the max-plus inner product, a process that produces the individual components of the updated vector. This vector must then be assembled, so that copies can be passed to the parallel inner product operations.

$$\begin{tikzcd}
x_1 = B_{11} + a_1 \arrow[r] & x_{2,1} = \max\{ x_1 , x_2 \} \arrow[r] & x_{3,1} = \max\{ x_{2,1} , x_{2,2} \} \arrow[r] & x_{4,1} = \max\{ x_{3,1} , x_{3,2} \} \\
x_2 = B_{12} + a_2 \arrow[ur] & & & \\
x_3 = B_{13} + a_3 \arrow[r] & x_{2,2} = \max\{ x_3 , x_4 \} \arrow[uur] & & \\
x_4 = B_{14} + a_4 \arrow[ur] & & & \\
x_5 = B_{15} + a_5 \arrow[r] & x_{2,3} = \max\{ x_5 , x_6 \} \arrow[r] & x_{3,2} = \max\{ x_{2,3} , x_{2,4} \} \arrow[uuuur] & \\
x_6 = B_{16} + a_6 \arrow[ur] & & & \\
x_7 = B_{17} + a_7 \arrow[r] & x_{2,4} = \max\{ x_7 , x_8 \} \arrow[uur] & & \\
x_8 = B_{18} + a_8 \arrow[ur] & & &
\end{tikzcd}$$

Figure 6.5 First row of max-plus matrix multiply

For each row of the *B* matrix, the tournament illustrated in Figure 6.5 must be processed. Parallelism exists at multiple layers: row-wise in *B,* column-wise in the addition of the vector a, with a diminishing level of parallelism as the tournament winners propagate. Efficiency of the FPGA processing relies heavily on leveraging the problem structure and getting data to the appropriate units at the appropriate times. We pipeline these row-wise inner products using the parallel nature of the FPGA.

The FPGA MVM has been constructed in VHDL to capitalize on the structure of Figure 6.5. The tournament is executed with a sequence of max and add operations, in a circuit structure illustrated in Figure 6.6. The index swapping stands out in this architecture. As a Matlab or C code, this algorithm is relatively simple to instantiate. When examined in VHDL, getting memory address signals that jump between registers is a complicated task that adds to VHDL-to-circuit compilation difficulties given the timing constraints. The problem is one of moving data back upstream in the physical flow through the circuit.

While this process seems straightforward, we note that communication with the DDR3 memory be addressed all the way down into the pipelined math core. The fundamental concept of DDR3 is that it has variable latencies due to the necessity to refresh its contents periodically as well as delays necessary for its own pipelines to function. This concept of wait permeates every decision in the Iteration Core.

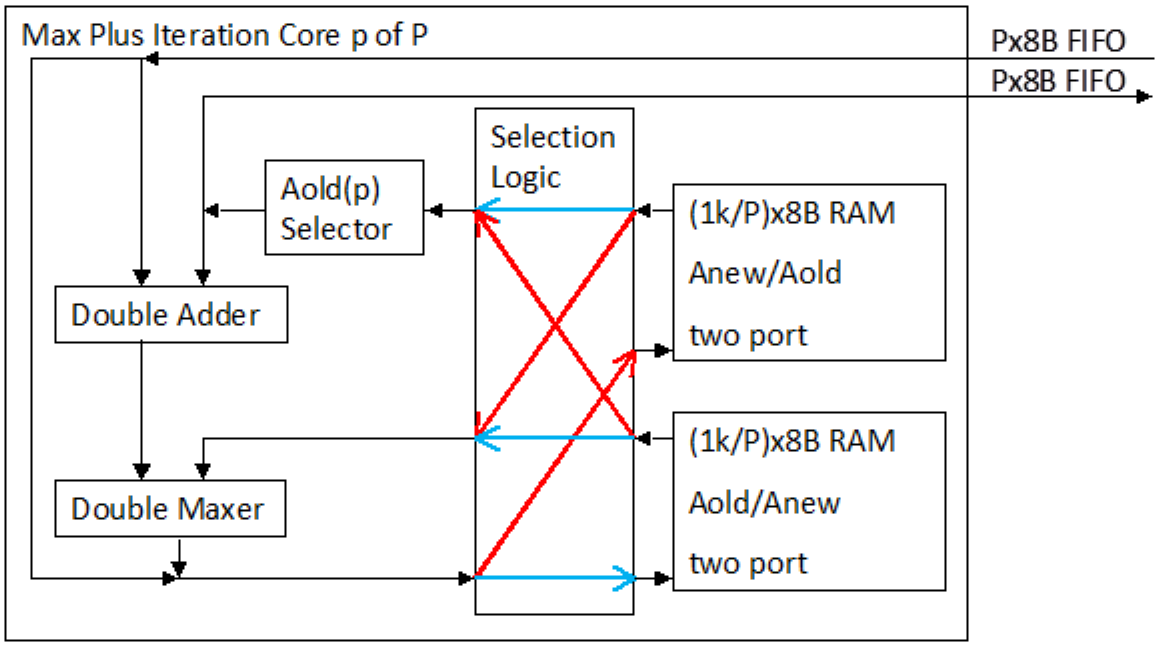

Max Plus Iteration Core p of P
Px8B FIFO
Px8B FIFO
Selection Logic
Aold(p) Selector
Double Adder
Double Maxer
(1k/P)x8B RAM Anew/Aold two port
(1k/P)x8B RAM Aold/Anew two port

Figure 6.6 Max-Plus Iteration Core organization

The architecture of the full system is of course dependent on the algorithm chosen for the max-plus matrix iteration, but in the context of a tool, the FPGA solution needs to interface with other new and preexisting MATLAB code sets, requiring some MEX writing to connect the C and FPGA with the user interface. That is, we need multiple components: several pieces of code run on the host computer to accept user input, to conduct some complex computations more suited to standard processors to construct the B matrix, and provide graphical output and data storage for controllers. Thus, the software/hardware partnership involves a number of host/FPGA design and load balancing considerations.

Interfacing the PC with the FPGA is a complex task in its own right. The PC interface needs to communicate data to the DDR3 memory on board the Xilinx platform, and the VHDL circuit needs to access the DDR3 memory. AVHDL state machine is necessary to manage the host computer transfers, DDR3 memory communication with the max-plus core, and the FPGA RAM operation. All of these requirements together comprise the circuit architecture developed below in Figure 6.7.

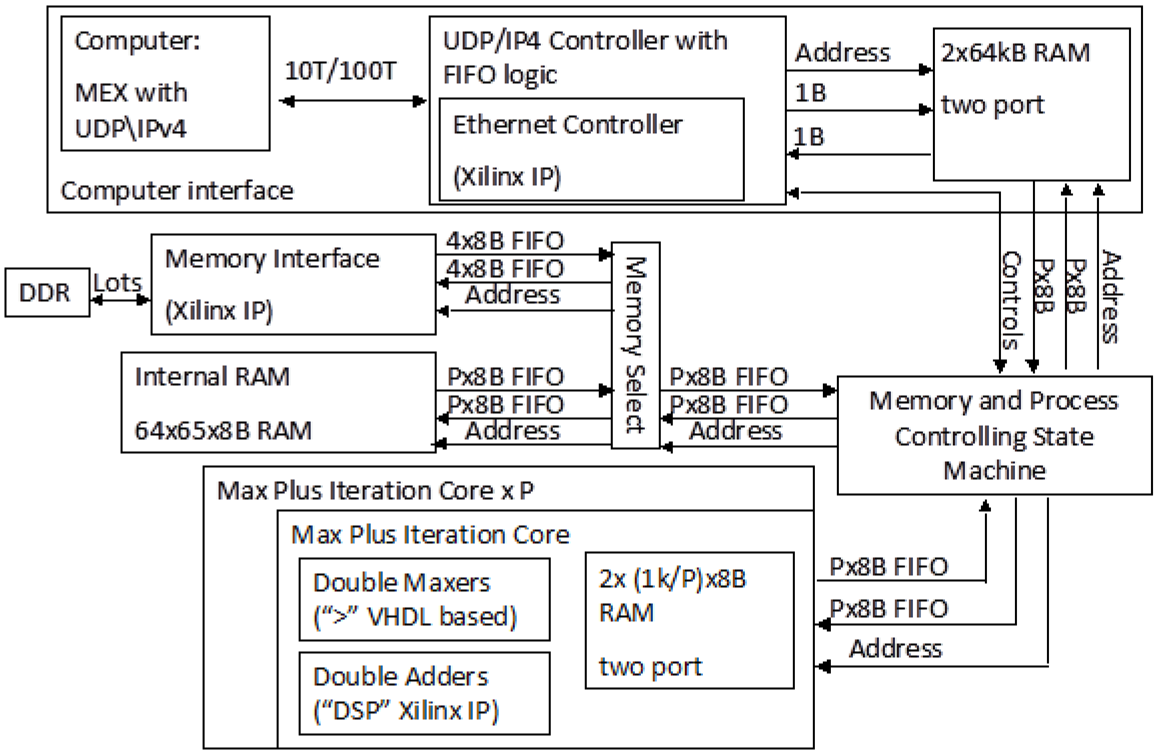

Computer:
MEX with UDP\IPv4
Computer interface
10T/100T
UDP/IP4 Controller with FIFO logic
Ethernet Controller (Xilinx IP)
Address
1B
1B
2x64kB RAM two port
Controls
Px8B
Px8B
Address
DDR
Lots
Memory Interface (Xilinx IP)
4x8B FIFO
4x8B FIFO
Address
Memory Select
Internal RAM 64x65x8B RAM
Px8B FIFO
Px8B FIFO
Address
Px8B FIFO
Px8B FIFO
Address
Memory and Process Controlling State Machine
Max Plus Iteration Core x P
Max Plus Iteration Core
Double Maxers (">" VHDL based)
Double Adders ("DSP" Xilinx IP)
2x (1k/P)x8B RAM two port
Px8B FIFO
Px8B FIFO
Address

Figure 6.7 Architecture of the full host-coprocessor system

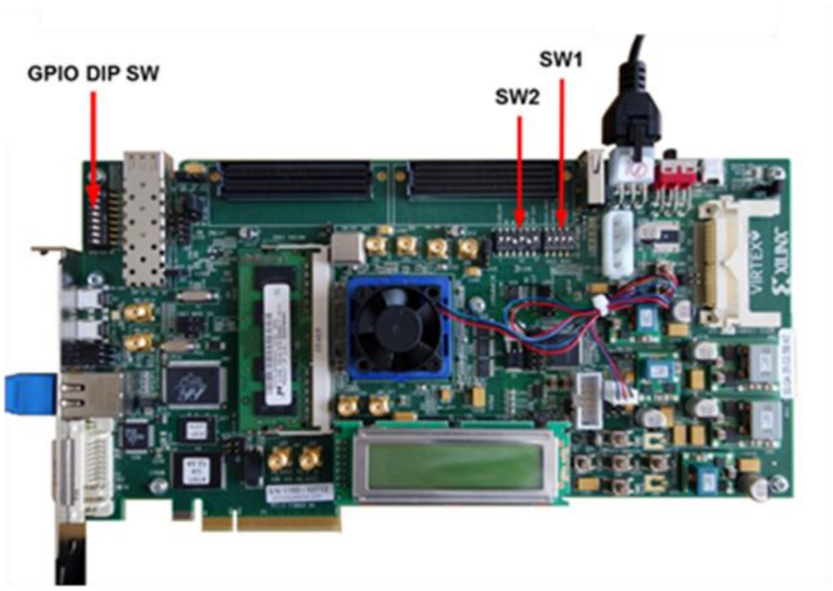
The finite-state machine controller includes the following basic processes: Write to Memory, Read From Memory, Start Max-Plus Processing, and Poll Max-Plus Status.

The primary challenge in creating the FPGA MVM is data transfer, access, and timing with the DDR3. The use of DDR3 on board the Xilinx platform is crucial for efficient computation due to the limited size of the FPGA internal RAM. Timing within the contest of DDR3-FPGA collaboration constrains us to under-clock to 250MHz, when 300-400MHz is theoretically possible.

6.3 Running the system

The following steps are a guide to connecting the ML605 and performing calculations on the Max-Plus MVM FPGA board.

1. Make sure all switches, SW1, SW2 and GPIO DIP SW.

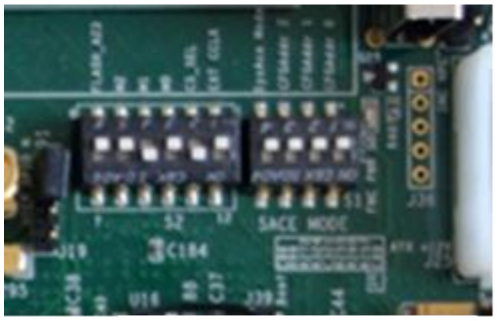


Switches SW1 and SW2 are next to each other:

SW1(1:4) = XXX0

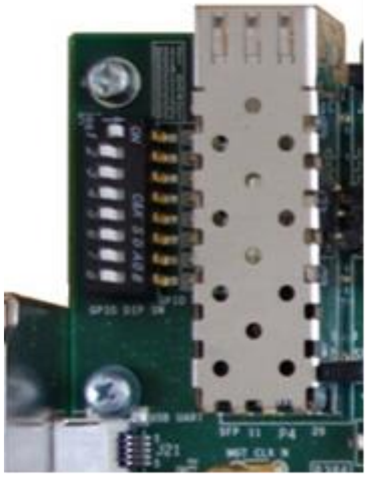
SW2(1:6) = 010100

The X means either 0 or 1. The first position of switches starts from right in the pictures.

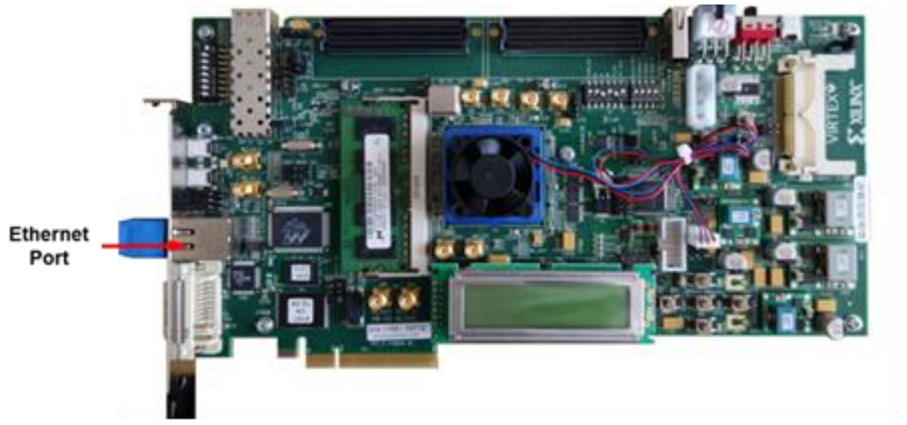


GPIO DIP SW(1:8) = 1XXXXXXX

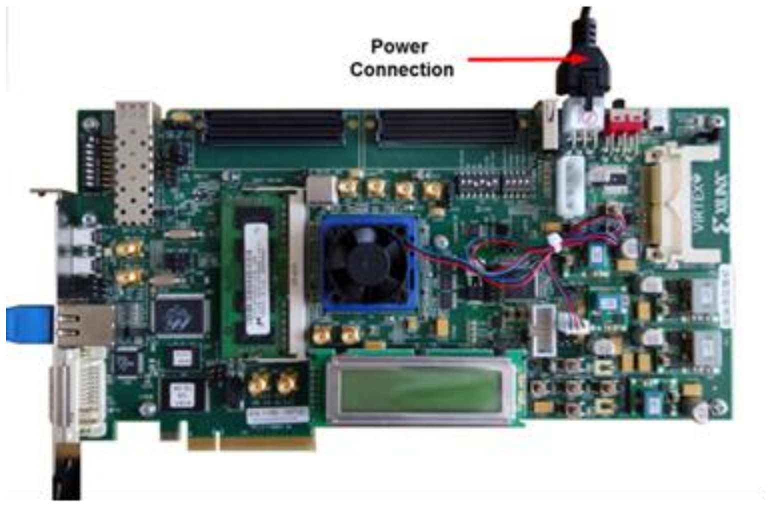
Here the first position of the switch is at the top in the picture.



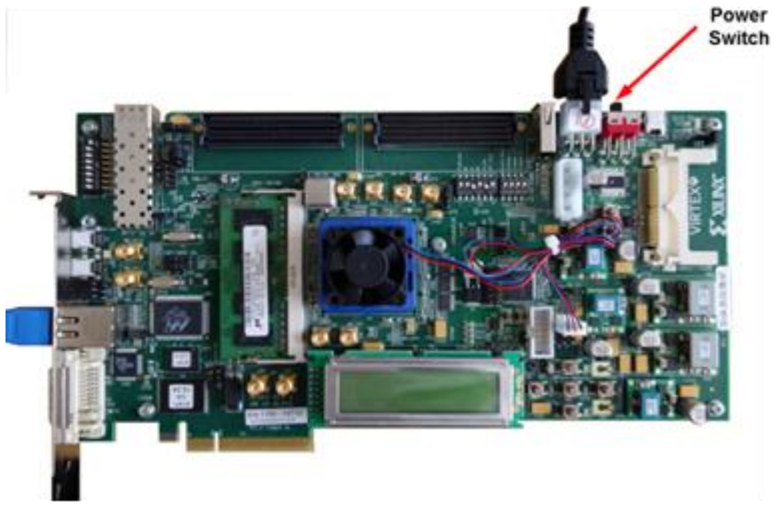
2. Connect Ethernet cable to PC and to ML605.



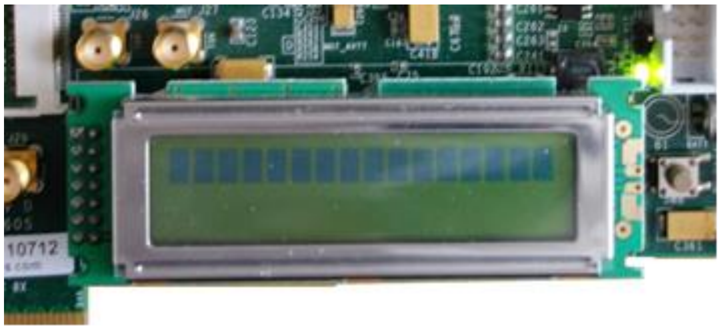
3. Connect power supply to ML605. The power connection is located on top, towards the right of the board.



4. Turn on the power to the ML605.

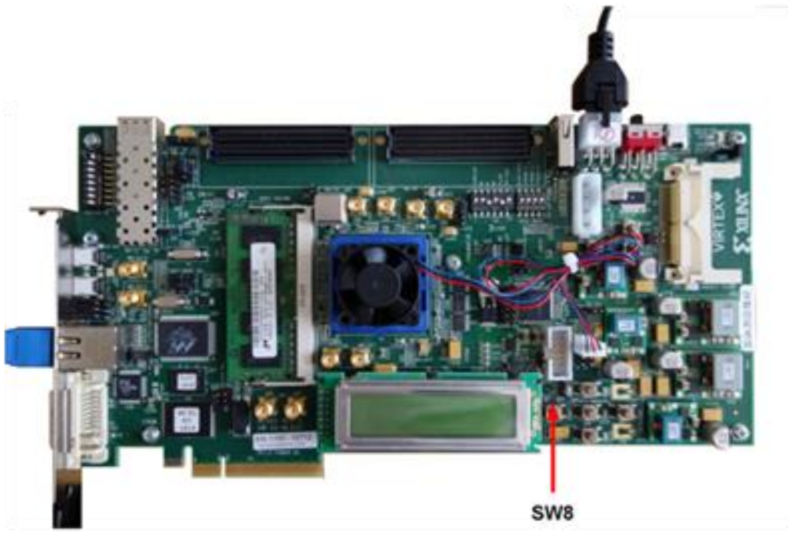


The LCD will initially have black boxes across the top.



5. Wait for the board to load the max-plus design from the onboard BPI flash into the FPGA (approximately 1 minute). When the FPGA has been loaded the LCD will be cleared.

6. At this point, press the reset button SW8,



The LCD should be completely filled with two rows of numbers (possibly, but not necessarily, all zeros). If it is not, press the reset button again. (It may take several tries to achieve a full two rows of numbers.)

7. The Max-Plus MVM FPGA is now ready to communicate with the PC. The PC and Board IP addresses must be set to 1.2.3.9 and 1.2.3.5 respectively. At Linux prompt type the following commands to set them. (Enter the password associated with the account on this PC if prompted to do so.)

   ```
   $ sudo ifconfig eth0 1.2.3.9 netmask 255.255.255.0
   $ sudo arp -s 1.2.3.5 00:0a:35:11:22:33
   ```

   Alternatively, at the Matlab prompt run the following script:

   ```
   >> Set_IP_addresses
   ```

   Once communications have been established, there should be other numbers displayed on the LCD besides all zeros.

The FPGA is now ready for use.

6.4 Max-Plus FPGA MVM computation

The user must create a matrix *B*, vector a_T and number of iterations *N*. A parameter must be provided by the user to indicate whether Max-Plus MVM solver will run N iterations regardless if the convergence criterion is met before iteration reaches to *N*. The Max-Plus MVM is considered converged if the change in results from two consecutive iterations is 0.

The user also must check if the size of matrix *B* is multiple of 32 (if the size is smaller than 256) or multiple of 128 (if the size is bigger than 256). If the condition is not met, the user must expand the matrix B to the size that is multiple of 32 or 128 by padding the matrix with -inf, the max-plus zero. A Matlab function is provided to do padding.

Matlab is used to interface between the host PC and the Board. The following steps are a guide to carry out Max-Plus MVM $a_0 = B^N \otimes a_T$ on the FPGA board.

1. Set up the Max-Plus MVM FPGA System as described above.
2. Create or load matrix B and vector aT in Matlab session.
3. Check if the number of elements in aT is the same as number of columns of matrix B, and if matrix B is square.
4. Check the size of matrix B. If it is not a multiple of 32 (if the size is smaller than 256) or multiple of 128 (if the size is bigger than 256), run the Matlab script

   ```
   >> [Bpad, aTpad] = Pad_data(B, aT(:), m32_128);
   ```

   where m32_128 is either 32 or 128, a user provided parameter. The use of aT(:) (instead of aT) is to guarantee the column-vector convention.

5. Run the Max-Plus MVM FPGA controller function, maxPlusLoopFPGA7.mexglx:

   ```
   >> a0pad = maxPlusLoopFPGA7(Bpad, aTpad, N);
   ```

   The controller function terminates either number of iterations reaches to N or when a0pad (a0 padded with –inf) converges, whichever comes first. An alternative controller function, as maxPlusLoopFPGA6(.mexglx) is provided if a user does not want an earlier termination. The function call is the same.

6. Recover a0 from a0pad. In step 5, we assumed the user padded the original matrix B and vector aT. Therefore the resulting vector a0pad is a padded vector. The solution to the problem should have the same size aT. Recovering a0 from a0pad is done easily in Matlab:

   ```
   >> a0 = a0pad(size(aT));
   ```

The example is provided through a Matlab function, MaxPlus_FPGA_Example.m (hereafter referred to as the example), serving as a jumping off point for a user. The B matrix in the example is 8064 by 8064 with entry B(i,j) = -(x(i) + y(j))2 – x(i)2, x = (-4032, -4031, …, 4031) and y = (-4032, -4031, …, 4031). The vector aT contains 8064 zeros. We set number of iterations N = 1000. In this example, the function maxPlusLoopFPGA7 is used as controller function. The example code can be used a driver with the matrix B and vector aT replaced by data of interests. The Matlab script is listed below, it illustrates the steps required to perform an iterated max-plus matrix vector multiplication using this Max-Plus MVM system:

```
%
% Establish communication with Max-Plus MVM FPGA board
%
Set_IP_addresses;

%
% Set parameters
%
M=8064;                 % B matrix size MxM and aT vector size Mx1
N=1000;                 % Number of iterations to work towards solution
early_term=1;           % Terminate iterations if solution has converged
                        %    before end of iterations.

%
% Create B matrix
%
disp(['Creating ', num2str(M), ' x ', num2str(M), ' matrix and ',...
       num2str(M), ' x 1 vector.']);

[x y] = ndgrid(-M/2:M/2-1,-M/2:M/2-1);
B     = -(x+y).^2-x.^2;


%
% Create aT vector
%
aT = zeros(M,1);

%
% Add padding if M not multiple of 128 (or, at least, 32)
%
disp(['Checking matrix and vector to add padding if necessary.'])

[Bpad aTpad] = Pad_data(B, aT, 128);
clear B aT;            % B and aT variables are no longer needed so
                       % free up the memory
%
% Run computations
%
disp(['Sending matrix to FPGA board and running computations.']);
```

```
tic;
if early_term == 1
    a0pad = maxPlusLoopFPGA7(Bpad,aTpad,N);
else
    a0pad = maxPlusLoopFPGA6(Bpad,aTpad,N);
end
etime = toc;
disp(['FPGA process elapsed time is ', num2str(etime), ' seconds.']);
%
% Remove any padding that may have been added prior to computations
%
a0 = a0pad(1:M,1);
```

The following is output from running the example. Note that there were a few “data (byte) problem only: -1 of 1024” messages before it settled down and processing began. This is not abnormal. This solution converged at iteration 15. If it was allowed to run through the 1000 iterations it would have taken more than three times longer.

```
Setting computer IP address to 1.2.3.9.
Setting FPGA IP address to 1.2.3.5.
Creating 8064 x 8064 matrix and 8064 x 1 vector.
Checking matrix and vector to add padding if necessary.
Sending matrix to FPGA board and running computations.
N:8064,I:1000,B11:-81285120.000000,Aold1:0.000000
enter maxPlus
setting up send to FPGA socket
setting up recieve from FPGA socket
masking 0 bits for percentDeltaAlert
sending Aold
data (byte) problem only: -1 of 1024
sending B
data (byte) problem only: -1 of 1024
sending Start Processing
masking 0 bits for percentDeltaAlert
sending Poll Processing
start poll
current poll : iteration = 65535  i = 0  j = 6792  latency = 0  converged = 65535
current poll : iteration = 0  i = 616  j = 7928  latency = 0  converged = 65535
current poll : iteration = 0  i = 1231  j = 7312  latency = 0  converged = 65535
…
current poll : iteration = 15  i = 7470  j = 4880  latency = 0  converged = 65535
current poll : iteration = 16  i = 21  j = 5736  latency = 0  converged = 15
Converged at 15, stopping early
current poll : iteration = 16  i = 637  j = 704  latency = 0  converged = 15
…
current poll : iteration = 17  i = 7947  j = 3368  latency = 0  converged = 15
current poll : iteration = 19  i = 0  j = 0  latency = 0  converged = 15
Iterations terminated.
sending read Anew
first Anew:-26304421.000000
FPGA process elapsed time is 49.4027 seconds.
```

For a size of 8,064 problem, it took the Board 15 iterations and total of 49.4 seconds.

When processing is complete shut down as follows:

1. Shut down Matlab so that it does not continue to communicate with the board.
2. Turn off the power to the ML605.Detach the power cable.
3. Detach the Ethernet cable.
4. Return the board to the static-resistant bag. Store the board in this bag when not in use.

6.5 Timing results

We have run an extensive suite of test problems through the FPGA implementation at 250Mhz and 311Mhz. The latter represents the fastest rate at which we could reliably drive the board. We compared this code to a Matlab implementation in which the max-plus matrix iteration was implemented in C and accessed through Matlab's MEX interface. This represents our most efficient purely-host-processor implementation. Figure 6.8 shows the results of these tests in terms of the speed per single time step of the max-plus dynamic programming computation, as instantiated in the max-plus arithmetic operation. One can see that the FPGA board begins to offer significant time savings for larger problems. Since it is common for higher dimensional problems to require thousands of basis elements, we see marked improvements to be had with this hardware/software partnership.

There are clear advantages to the parallelism and hardware-instantiated max-plus arithmetic that the FPGA provides. The prototype still contains a number of opportunities for improvement, but we have developed a design that is protectable as intellectual property and offers a direct path to ASIC development and are seeking Phase III partnerships for such.

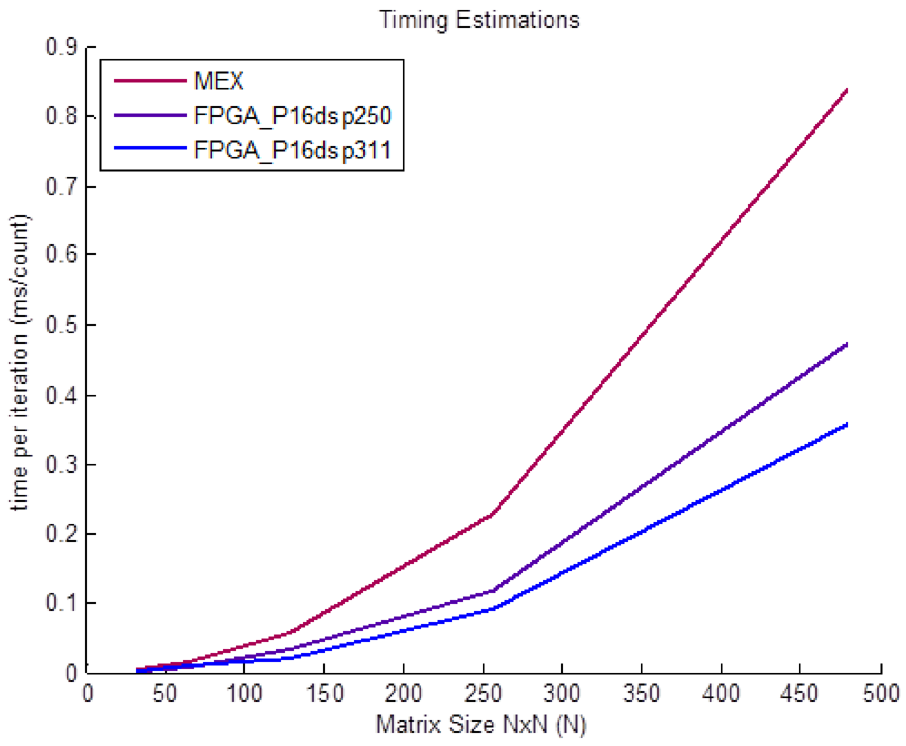


Figure 6.8 Comparison of host processor (MEX) and FPGA max-plus matrix iteration speed

7.0 <u>The Linear Quadratic Regulator</u>

The linear-quadratic regulator (LQR) is among the best known optimal control problems. With a well-understood solution that is easily computable, the LQR problem provides an excellent testing platform.

The linear dynamical system

$$\dot{x} = Ax + Bu\,, \tag{7.1}$$

in which $x = x(t)$ is the n-dimensional system state, $u = u(t)$ is the m-dimensional system state, and A and B are appropriately dimensioned matrices, defines the LQR dynamics. The quadratic cost is given by

$$J(u; y, t_0, t_f) = -\frac{1}{2}\int_{t_0}^{t_1} x^T Qx + u^T Ru\, dt\,, \tag{7.2}$$

in which we use a minus for maximization purposes.

The value function is given by the quadratic form

$$V(x,t) = x^T P(t) x, \tag{7.3}$$

in which *P* satisfies the well-known Riccati differential equation

$$\dot{P}(t) + A^T P(t) + P(t)A + Q - P(t)BR^{-1}B^T P(t) = 0, \quad P(t_f) = 0, \tag{7.4}$$

and the optimal control and state are given by

$$\begin{aligned} u^*(t) &= -R^{-1}B^T P(t) x^*(t) \\ \dot{x}^*(t) &= \left(A - BR^{-1}B^T P(t)\right) x^*(t) \end{aligned} . \tag{7.5}$$

To develop some analytical insight into the max-plus time stepping, we consider an LQR formulation with a terminal quadratic cost over a short time interval (*t*, *t*+*h*):

$$\begin{aligned} &\dot{x} = Ax + Bu, \quad x(t) = y \\ &J(u) = -\frac{1}{2}\int_t^{t+h} x^T Q x + u^T R u \, dt - \frac{c}{2}\left|x(t+h) - x_i\right|^2, \end{aligned} \tag{7.6}$$

to which we will apply max-plus computation using quadratic basis elements. The resulting value function requires two matrices:

$$\begin{aligned} &\dot{P} + A^T P + PA + Q - PBR^{-1}B^T P = 0, \quad P(s) = cI \\ &\dot{W} + A^T W - PBR^{-1}B^T W = 0, \quad W(s) = -cX = -c[x_1, x_2, \cdots, x_N] \end{aligned} . \tag{7.7}$$

The optimal control is given by for the subproblem (7.6), and the value function is given by

$$V(t,y) = \frac{1}{2} y^T P(t) y + \frac{1}{2} y^T W(t) e_i - \frac{c}{2} x(s)^T x_i + \frac{c}{2} x_i^{\,T} x_i = S_{t,s}(\psi_i)(y). \tag{7.8}$$

By considering the LQR problem, then, we can test the accuracy of approximations to computing the B matrix and the M matrix from the mixed finite element method as well as those for computing the value function and optimal control.

As a simple illustrative problem for validating computations, we consider a 1-d problem with *A*=*B*=*Q*=*R*=1. We take the final time of 10. Figure 7.1 shows the analytically computed *B* and *M* matrices of Equations (4.10) and (4.13) for basis and test function grids of size 128 on the state space range of [-2, 2], while Figure 7.2 shows the numerically computed matrices.

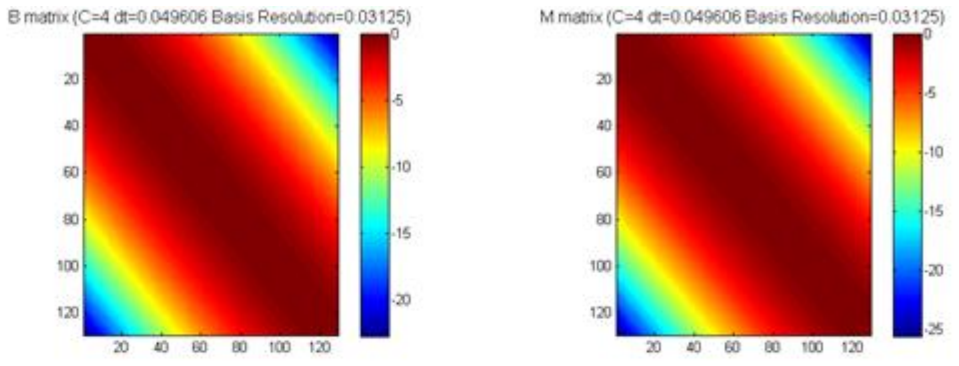


Figure 7.1 Analytically computed B and M matrices for a 1-D LQR problem: left panel shows the B matrix and right panel shows the M matrix

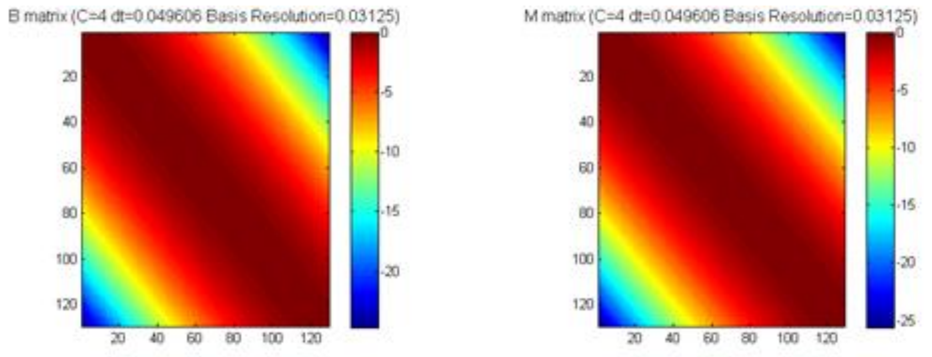


Figure 7.2 Numerically computed B and M matrices for a 1-D LQR problem: left panel shows the B matrix and right panel shows the M matrix

Next, we compare the analytically computed value function (based on the Riccati solution of Equations 7.6-7.7) with that determined by the max-plus finite element computation in Figure 7.3.

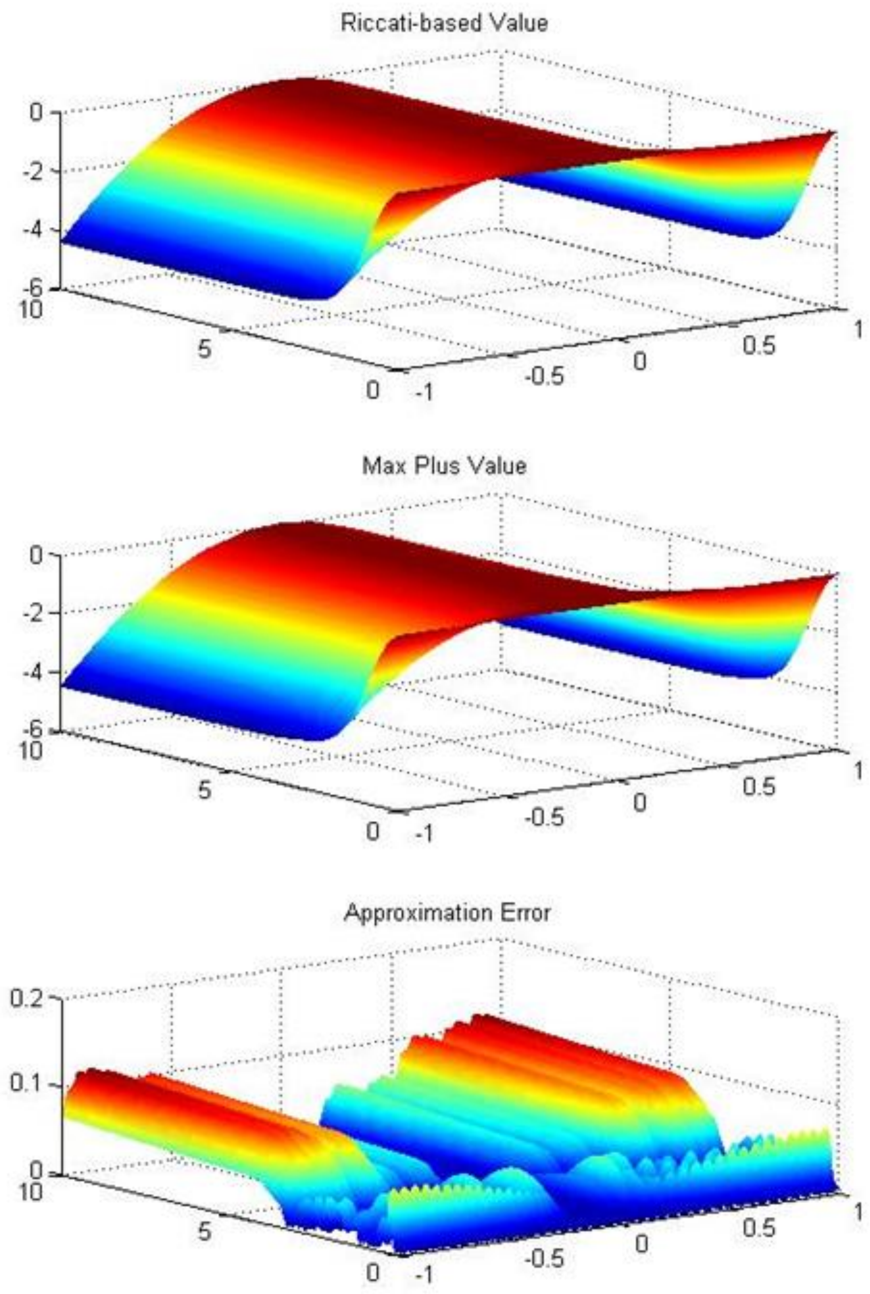


Figure 7.3 Analytic and numeric value functions, together with their difference

Two heat maps are given, for a coarse grid of Δx = 0.2, on the left, and a finer grid of Δx = 0.04, on the right. Numerical experience suggests that a shape parameter that is 4-8 times the Riccati matrix will work quite well.

In Figure 7.4, the upper envelope highlighted with a white curve gives *c* as equal to the Riccati solution across the 41 LQR problems. One can easily see that a shape parameter smaller that the

Riccati solution produces large error. This result is somewhat intuitive, as the quadratic finite elements are too “wide” to fit “inside” the value function when *c* is too small. One can also see that the area of small error is much more forgiving in the case of the finer grid.

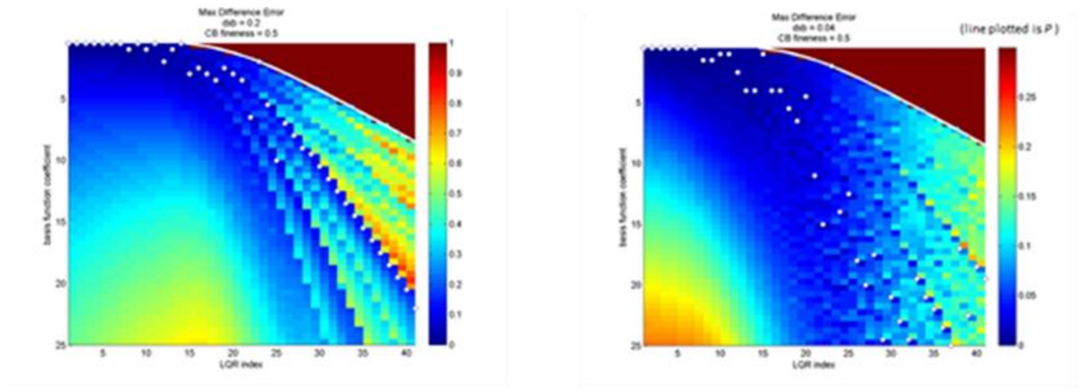


Figure 7.4 Approximation error heat maps for coarse and fine grid approximations

8.0 <u>Perimeter Patrol Planning – An Example</u>

The use of and interest in unmanned air vehicles (UAVs) as important components in military operations continues to grow. The US Air Force’s UAV plan [DS2047] indicates a goal of moving from “man in the loop” to “man on the loop,” requiring a fairly high level of autonomous operation. Significant improvements in autonomy are necessary to achieve this ambitious program, including appropriate concepts of operation, UAV design, computational algorithms, communications, and human/machine interfaces. Of great interest is pushing the envelope of current capability to allow unmanned autonomous systems to make time-critical decisions without the need for human oversight or control.

Such levels of autonomy for UAVs are a serious challenge for computational technology. In addition to the complexity of algorithms needed to make decisions computationally using battlespace feedback in the presence of uncertainty, creating autonomous systems that work well with men on the loop is a significant challenge.

Humans tend to make decisions in a manner that is quite different from computational systems, leading to issues of trust [Ke1]. Generally this trust problem arises from the human’s inability to understand why the algorithm selected the approach that it did. The computational choice appears as nothing more than an authoritative pronouncement, and the operator has no means of assessing its reliability or effectiveness a priori.

Among the more promising avenues to increased autonomy are mission planning problems that balance the strengths and weaknesses of autonomous computationally driven task components and man on the loop human decision making. Assigning tasks to the autonomy that can be understood, at least qualitatively if not quantitatively, offers an approach with potential for success.

Additional issues in autonomous UAV tasking involve computational and communication limitations for smaller, more cost-effective vehicles. With the computational ability to identify and track targets in the battlespace without operator command requirements, UAVs could conduct a number of advanced intelligence and surveillance tasks at a high level of autonomy. Current systems do not have such capabilities, and humans must observe the visual data that streams from UAVs, detecting targets of interest and directing UAV paths. This concept of operations does little to relieve the human workload.

There are, however, related concepts that allow certain levels of autonomy coupled with appropriate human operation. One such approach, the Air Force Research Laboratory's Cooperative Operations in UrbaN TERrain (COUNTER) project described in [GRCF, CHHDP], involves a networked group of unmanned ground sensors (UGSs). These UGSs are positioned in the battlespace at locations of interest (which we call stations), and they transmit alerts when they detect a potential target in their field of regard. These sensors may be seismic, chemical, or some other relatively inexpensive and simple technology. When an UGS puts an alert onto the network, the UAVs can then travel to and inspect the station by capturing video. Depending on the capabilities of the UAV and the network, the video may be transmitted via the network or delivered physically to a human for examination. The COUNTER subproblem we consider is one of a collection of UGSs stationed around the perimeter of a protected area. The UGSs generate alerts signaling a potential incursion attempt into the protected region. The UAVs patrol this perimeter, searching stations at which UGSs have generated alerts. This form of human/UAV collaboration balances nicely the abilities of computational autonomy and operator intervention. The decisions of where to send and when to retask are handled by the automation, while the human focuses on the visual imagery to detect intrusions.

8.1 Perimeter Patrol Problem Definition

Our model begins with m UGSs and n UAVs. The UGSs are positioned at fixed locations on the perimeter of a protected region, which is modeled as a simple closed curve surrounding the area of interest. The curve is assumed to be parameterized in polar coordinates $(r(\theta),\theta), \theta \in [0,2p)$.

The arrival rate of alarms follows a Poisson process with rate α. We assume that at any given time, at most one station can receive an alert. An alert arrival is equally likely to occur at any of the stations occupied by UGSs, leading to an effective individual arrival rate of α/m. We model this process with indicator variables of alert occurrence. That is, we denote by $Y(t) = (Y_1(t), Y_2(t), \ldots, Y_m(t))$ the vector of alerts that occur at time t. By assumption, at most one entry in the vector can be 1. If all are zero, no alert occurs at time t.

This perimeter patrol problem is illustrated in Figure 8.1. UAVs follow the path around the area of interest, visiting UGS stations that generate alerts. The circles denote UGSs, with the red filled circle denoting an alarmed state. The UAVs fly patrol along the perimeter. When a UAV arrives at an alarmed UGS, the UAV's sensor is engaged so that it can investigate the UGS's vicinity.

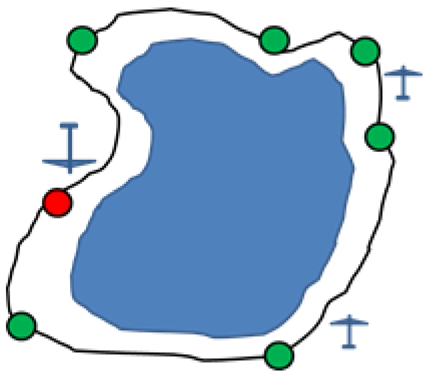


Figure 8.1 Perimeter patrol illustration

An additional step in the modeling effort is the discretization of the continuous perimeter into N+1 discrete locations, 0, 1, …, N. The distance between any two locations is computed taking into account the wrapping:

$$dist(i,j) = \min\{|i-j|, N-|i-j|\}. \tag{8.1}$$

The stations, by which we mean the locations at which the UGSs reside, are denoted ($g_1,\ldots,g_m$) $\in S=\{0,1,\ldots,N\}$. These are fixed problem parameters. The locations of the UAVs, denoted by $V = (V_1,\ldots,V_n) \in S$, are state variables. The dynamics for this component of the state are given by

$$v(t+h) = v(t) + u(t). \tag{8.2}$$

Here h denotes the time step required to move a UAV from one location in the state space to an adjacent location. The control variable u satisfies the constraint $u \in \{-1, 0, 1\}$, in which -1 and 1 denote movement to an adjacent location. The control action $u = 0$ denotes a loiter move for the UAV, which is allowed only when the UAV is at a station $v_i(t) = g_j$ for some j. Additionally, we constrain the UAVs so that they only loiter at UGS positions that have generated alerts. Thus, we need an additional state variable to keep up with the alert state.

In [KPDC], a model of alerts involves a running indicator A_i(t) governed by the dynamic equation

$$A_i(t+h)\begin{cases} 0 & v_k(t)=i, u_k(t)=0 \\ A_i(t) \vee Y_i(t), & \text{otherwise} \end{cases}. \tag{8.3}$$

The symbol ∨ denotes Boolean "or." Note that the alarm state is zeroed out upon a loiter by one of the UAVs. In addition the model of [KPDC] keeps up with the length of time each alert waits to be serviced. We denote by $T_i(t)$ the number of time steps an alert at station i has been waiting to be serviced by a UAV. This state variable has dynamics

$$T_i(t+h) = \min\{A_i(t)(T_i(t)+1), \overline{T}\}, \tag{8.4}$$

in which $\overline{T}$ denotes the maximum wait time, another parameter to be specified. Finally, we define a dwell state

$$d(t+h) = \left(d(t)+1\right)\left(1-\left|u(t)\right|\right), \tag{8.5}$$

to keep up with the length of time the UAV spends at a given station.

Taking the position, alert status, waiting times, and dwell times together as the system state *x*, we have a dynamical system that can be written as

$$x(t+h) = f(x(t), u(t), Y(t)), \tag{8.6}$$

in which *f* encapsulated the dynamics, u is the control, and *Y* is the noise process. With the dynamics specified, we turn to the control problem.

The optimization posed in [KPDC] takes a quality of service approach. The cost function implemented therein balances information gained by loitering against the waiting times of alerts. We define the objective functional *J* by

$$J(u) = E\left[\sum_{k=1}^{\infty} \lambda^k \Big(I(d(kh))(1-|u(kh)|) - \beta \max_i T_i(kh) \Big)\right], \tag{8.7}$$

an infinite time horizon discounted reward structure. In this cost functional, parameters λ and β denote the discount factor and penalty weight, respectively. The function *I*(*d*) denotes the information obtained by loitering for *d* dwell periods, forming the reward component. The penalty component (a negative reward or cost) is a scaled maximum waiting time: we seek a patrolling and dwelling strategy that avoids long waits.

The information function is meant to model an operator's ability to extract information from the data the UAV collects. In order to task the UAVs autonomously, the UAV needs to have a model that describes what is gained by loitering. The information model is described in detail in [CHHDP], treating the operator as a sensor with a confusion matrix with correct and false detection of threats and nuisances. The Shannon information can be computed as a function of the number of dwells:

$$
\begin{aligned}
I(d) = {} & pP_{TR}(d)\log\left(\frac{P_{TR}(d)}{pP_{TR}(d) + (1-p)(1-P_{NR}(d))}\right) \\
& + p(1-P_{TR}(d))\log\left(\frac{1-P_{TR}(d)}{p(1-P_{TR}(d)) + (1-p)P_{NR}(d)}\right) \\
& + (1-p)(1-P_{NR}(d))\log\left(\frac{1-P_{NR}(d)}{pP_{TR}(d) + (1-p)(1-P_{NR}(d))}\right) \\
& + (1-p)P_{NR}(d)\log\left(\frac{P_{NR}(d)}{p(1-P_{TR}(d)) + (1-p)P_{NR}(d)}\right)
\end{aligned}, \qquad (8.8)
$$

in which p denotes the a priori probability that an alert is a threat (not a nuisance), $P_{TR}(d)$ denotes the dwell dependent conditional likelihood of recognizing an existing threat, and $P_{NR}(d)$ denotes the dwell dependent conditional likelihood of recognizing an existing nuisance. The functions P_{TR} and P_{NR} both increase with d towards an asymptotic probability (that can be strictly less than one) so that there is a diminishing return for dwelling many times. The graph of an example information function, following the parametric choices of [CHHDP], is illustrated in Figure 8.2.

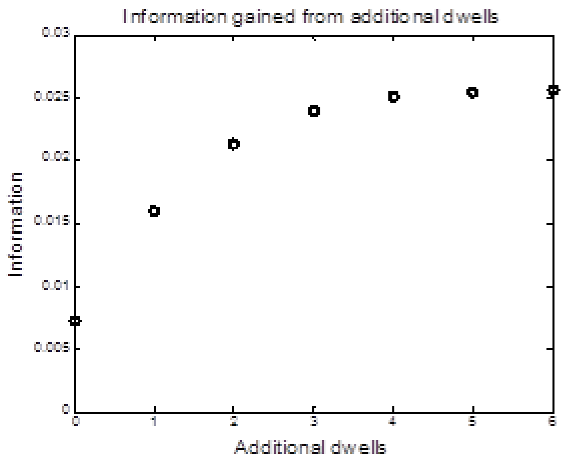


Figure 8.2 An example information gain function

With the information function in hand, we may turn to the problem of stochastic control. Maximizing the discounted infinite horizon reward J over UAV control plans attempts to increase information gain from dwells against the waiting times of unvisited stations. The infinite horizon discounted reward, coupled with bounded information function and wait times, leads to a time-independent value function and a full state feedback control.

The dynamic programming equation for this problem is given by

$$V(x_0) = \max_u \left\{ I(d)(1-|u|) - \beta \max_{1 \le k \le m} T_k + \sum_{k=1}^{m+1} p_k V(f(x_0, u, Y_k) \right\}, \tag{8.9}$$

in which the probabilities p_k denote the probability of arrival state Y_k, $k = 1,2,\ldots,m+1$. These probabilities are determined directly from the Poisson model of an event occurring with rate α and being assigned randomly to one of the UGS stations. The indices 1 through m denote an arrival at that station, and the index m+1 denotes no alert arrives in the time step.

Note that the state space for this problem can become quite large. With five UGSs stationed on the perimeter and a maximum wait time of 15 time steps, with 20 discrete spatial positions for a single UAV, the state space contains approximately 15^5x20 or over 15 million states, without even considering the dwell time state variable.

A leaner approach to optimization involves a different objective functional. Instead of information gain, we seek to maximize the probability that no alert evades inspection. Toward this end, we need some additional modeling to describe how an alarm is missed. If an UGS triggers an alert, it means there was an intrusion detection. Whether the intrusion was a threat or a nuisance, it must be investigated by the UAV. The longer the alert waits to be serviced, the greater the chance that a triggering threat will escape, evading detection and inspection by the UAV.

We model this process simply with a single step "dispersion" probability. We assume that the entity triggering the alert becomes undetectable by the UAV with probability q. The probability of a triggering entity remaining detectable for k time steps is $(1 - q)^k$. This probability captures the conditional likelihood that an existing alert is not missed. The full probability of not missing an alert is

$$\begin{aligned} G(x,u) &= \prod_{k=1}^{m} P[\text{an alarm at UGS}\,k \text{ is not missed}] \\ &= \prod_{k=1}^{m} \left[(1-q)^{T_k} I(T_k > 0) + \sum_{j=1}^{n} \left(1 - P_{TR}(d_j)\right) I(v_j = k) p + \sum_{j=1}^{n} \left(1 - P_{NR}(d_j)\right) I(v_j = k)(1-p) \right] \end{aligned}, \tag{8.10}$$

which includes the dispersion probability for a station that is waiting and the probability of missing a detection during dwells.

We make an approximation at this point, in order to reduce the state space. The waiting-time state variable is very expensive to maintain from a computational perspective, so we approximate the dispersion term of the missed alarm probability with the following:

$$\begin{aligned}(1-q)^{T_k} I(T_k > 0) &\approx (1-q)^M \\ M &= \min_i \{d_i + dist(v_i, k)\}\end{aligned} . \qquad (8.11)$$

This rather coarse waiting time approximation is really an approximation of the last visit time by one of the UAVs. The approximation has the effect on the optimization of keeping *M* small by reducing dwells or maintaining smaller distances between UAVs and UGSs. The other terms, modeling the information gain from dwelling, tend to keep dwell time high, attaining a similar balancing objective to that of the previous cost structure. The advantage to this approximation is that we no longer need to keep the waiting times as state variables, reducing the dimensionality dramatically.

8.2 A Single UAV

We begin with a comparison of our approach with that of [KPDC], in which (8.7) is optimized. We refer to this approach as the "Standard Solution" in the following discussion. In [KPDC], an alternative approach that approximates the standard solution with an aggregation scheme to reduce the number of states is considered, and we implement that approach as well, referring to it as "Aggregated Solution." Here we treat a single UAV checking on four UGSs, with a perimeter discretized with 15 locations. A maximum dwell, *d* = 5, and a maximum wait time, *w* = 15, are also used in [KPDC].

For comparison purposes, we must choose the max-plus probability parameters of missing alarm sources (*q* and *p*) in Equation (8.10). Figure 8.3 suggests that the best values for the max-plus probability solution to be *p* = *q* = 0.6. Our approach is denoted as "Max-Plus Probability Solution" in the following discussion.

Note that multiple solutions tested are practically tied with each other. To break the tie, this process was repeated for the dual UAV problem, and the intersection of the best solutions was used. There were still ties, but without a more stringent performance criteria, like dwell time distributions, the decision to use $p = q = 0.6$ was still somewhat arbitrary with respect to the given performance measures.

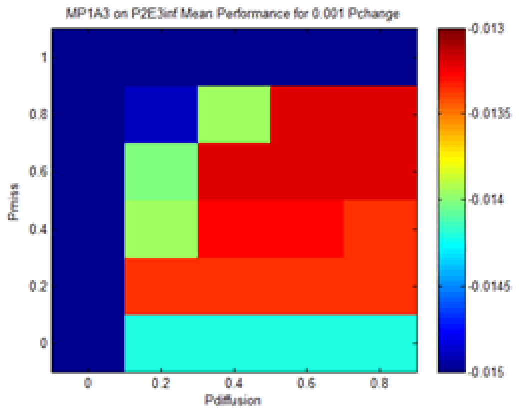


Figure 8.3 Max-Plus Probability Solutions Parameter Sweep

We first attempt to validate our implementation of [KPDC]. The Aggregated Solution is compared against the Standard Solution in Figure 8.4, for a sampling of initial states that index the x-axis.

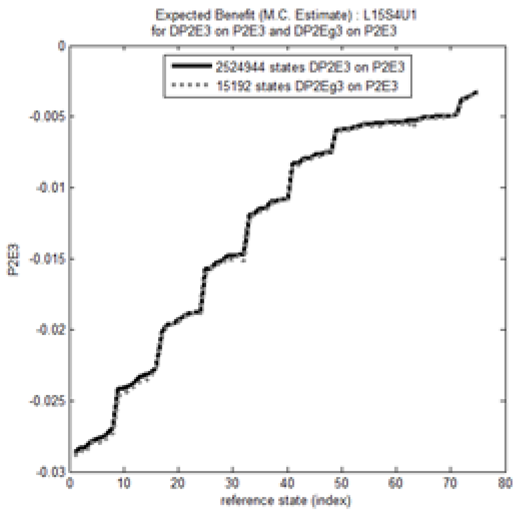


Figure 8.4 Aggregate Solution vs. Standard Solution

As shown by [KPDC] the Aggregated Solution has similar performance to the standard solution. For problems where the Standard Solution is impractical the Aggregated Solution will be used as a baseline if possible.

With the optimized parameters, the Max-Plus Probability Solution is compared against the standard solution in Figure 8.5. It is clear that the Max-Plus Probability Solution has similar performance to the Standard Solution.

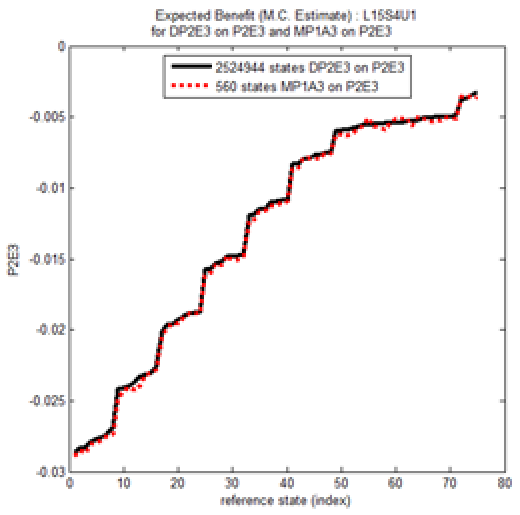


Figure 8.5 Max-Plus Probability Solution vs. Standard Solution

An important observable statistic is the dwell time. Since information is at the heart of the baseline optimality criterion, dwell time is a key component of the optimal reward.

The Aggregated Solution is compared against the Standard Solution (see Figure 8.6). The Max-Plus Probability Solution is compared against the Standard Solution in Figure 8.7.

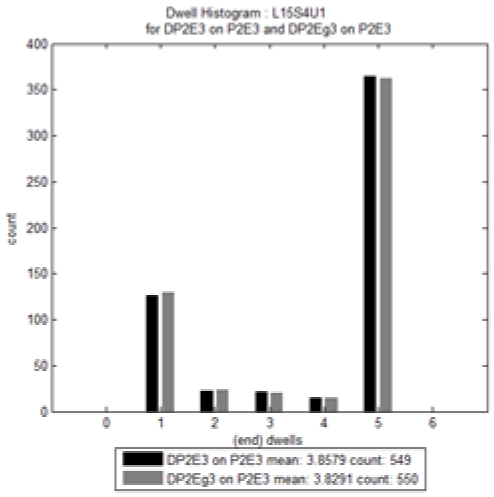


Figure 8.6 Aggregate Solution and Standard Solution Dwells

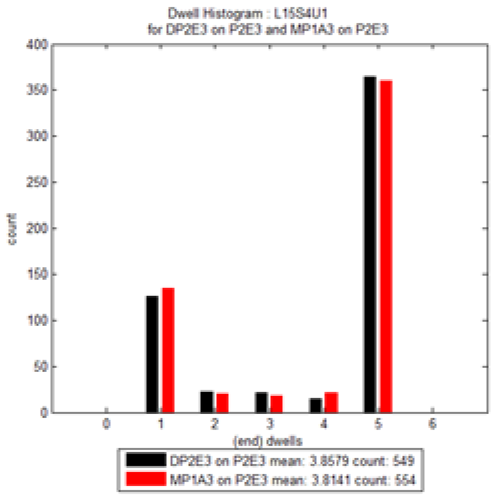


Figure 8.7 Max-Plus Probability Solution and Standard Solution Dwells

It should be noted for this problem that maximum dwell, d = 5, is optimal in the sense that the mean dwell time is converged for a sequence of standard solutions (see Figure 8.8). Note how the dwell distribution does not change after d = 5. The baseline used here is the Aggregate Solution as the Standard Solution is already into millions of states.

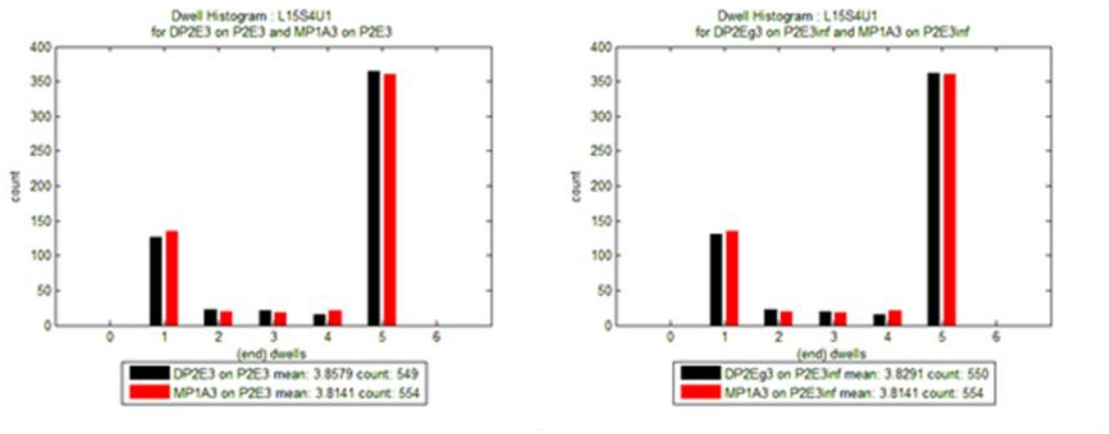


Figure 8.8 Dwell Behavior, Left d=5 Right d=6.

Wait times are also a crucial part of the problem. Higher wait times mean increased likelihood of a missed intrusion. While our max-plus model uses intrusion escape directly, the Standard Solution uses wait time penalty as a surrogate.

The Aggregated Solution is compared against the Standard Solution in Figure 8.9.

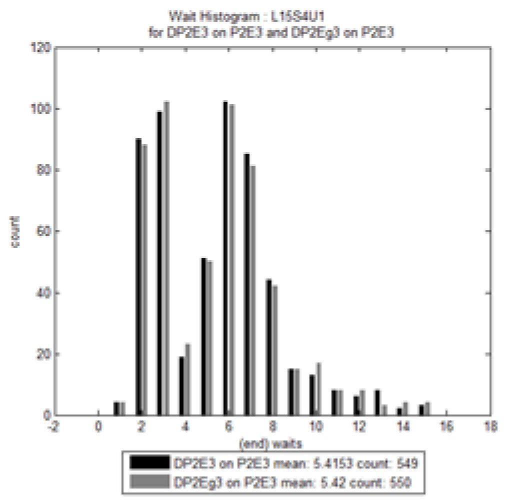


Figure 8.9 Aggregate Solution and Standard Solution Waits

The Max-Plus Probability Solution is compared against the Standard Solution in Figure 8.10. The mean wait times are similar. Technically the distributions are measurably different; this is due to the idle behavior where the Standard Solution hovers between the two stations that are near each other, waiting for one of them to generate an alarm, whereas the Max-Plus Probability

Solution idles by traversing the entire perimeter (consistent with the different objective criterion).

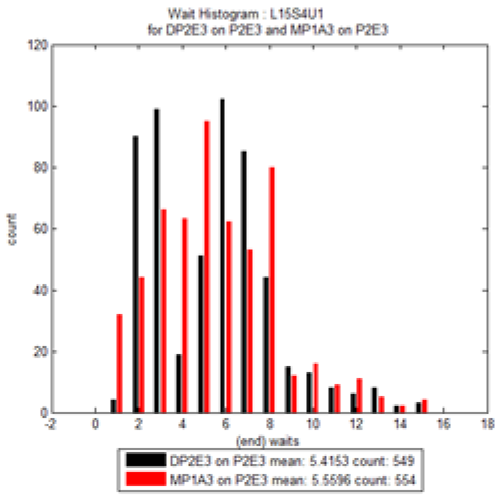


Figure 8.10 Max-Plus Probability Solution and Standard Solution Waits

Dwell times and wait times are state variables that drives the size of the state space. We investigated the upper bound on these states in Figure 8.11

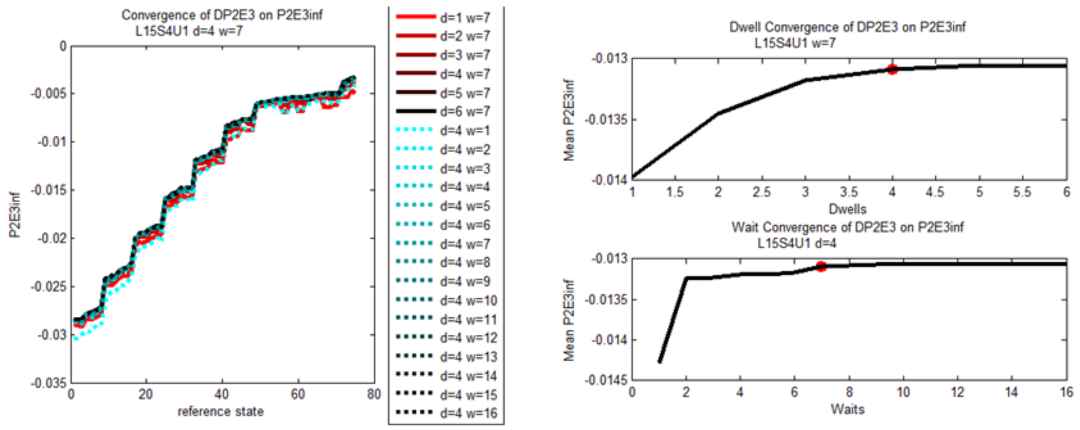


Figure 8.11 Left panel shows the optimal cost of the Standard Solution for a sampling of initial states under a variety of dwell/wait time upper bounds. Right panel shows cost as a function of the upper bounds on dwells (top right) and waits (bottom right)

Figure 8.12 shows the standard cost criterion behavior of the max-plus controller under a variety of dwell upper bounds. Since wait times are not a state in the max-plus formulation, the wait time upper bound is not relevant there.

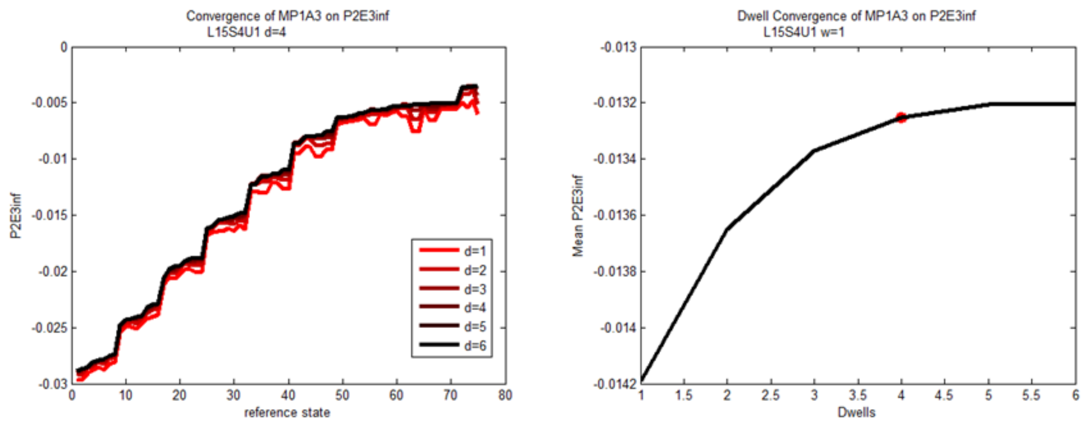


Figure 8.12 Convergence of Max-Plus Probability Solutions on the Standard Solution's performance measure

8.3 Two UAVs

The two UAV problem requires a significantly larger state space. In order to solve the problem under the Standard Solution approach, the maximum dwell and a maximum wait time need to be selected. Using convergence testing of the mean performance measure over a sample of initial states, Figure 8.13 indicates that performance is stable near $d = 5$ and $w = 3$. Note that these are the parameters that will be used in this section even though the Max-Plus Probability Solution converges with a different maximum dwell.

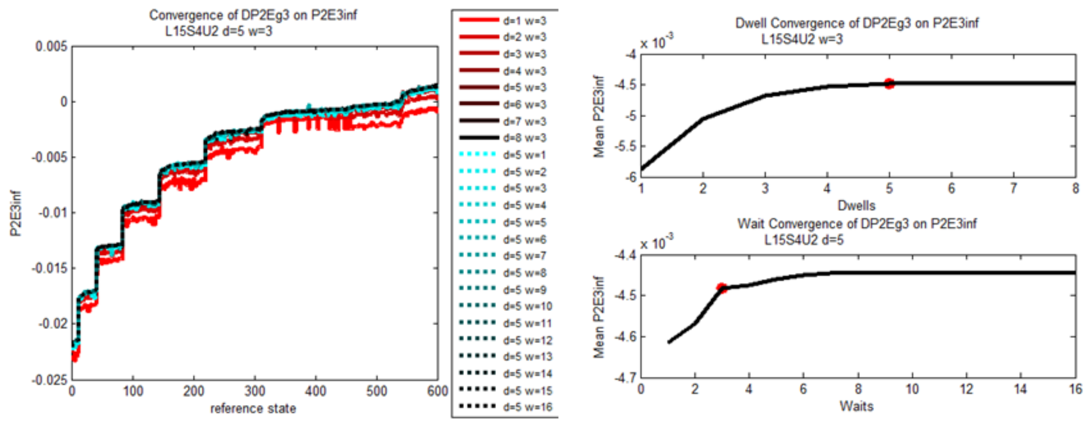


Figure 8.13 Left panel shows the optimal cost of the Standard Solution for a sampling of initial states under a variety of dwell/wait time upper bounds. Right panel shows cost as a function of the upper bounds on dwells (top right) and waits (bottom right)

Figure 8.14 shows the dependence of the cost functional on maximum dwell over a set of initialization states for those solutions.

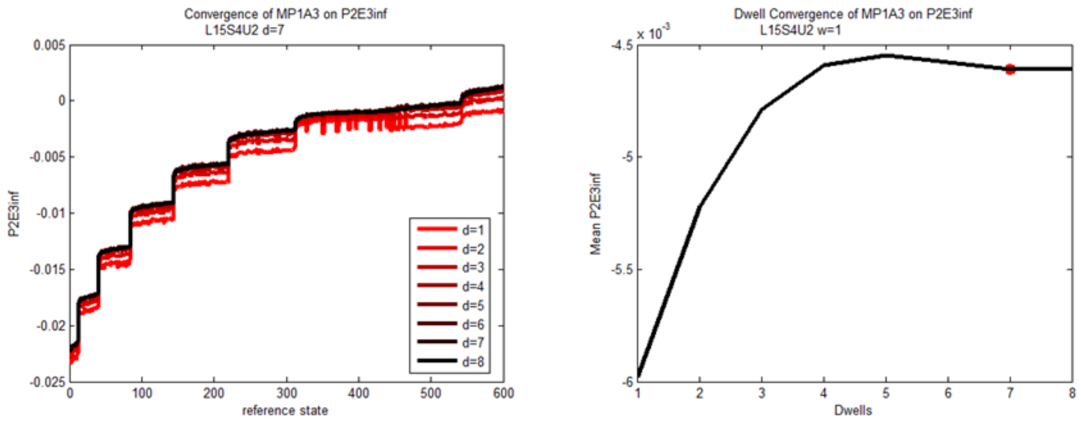


Figure 8.14 Dwell Dependence of Max-Plus Probability Solutions on the Standard Solution's performance measure

With respect to the performance measure dependence on dwell, an example of optimal maximum dwell time is clearly available in this problem. Recall the convergence of the max-plus probability solution with respect to dwell time. It was constrained to a maximum dwell of *d* = 5 for the performance conclusions but Figure 8.15 allows the maximum wait and dwell to increase.

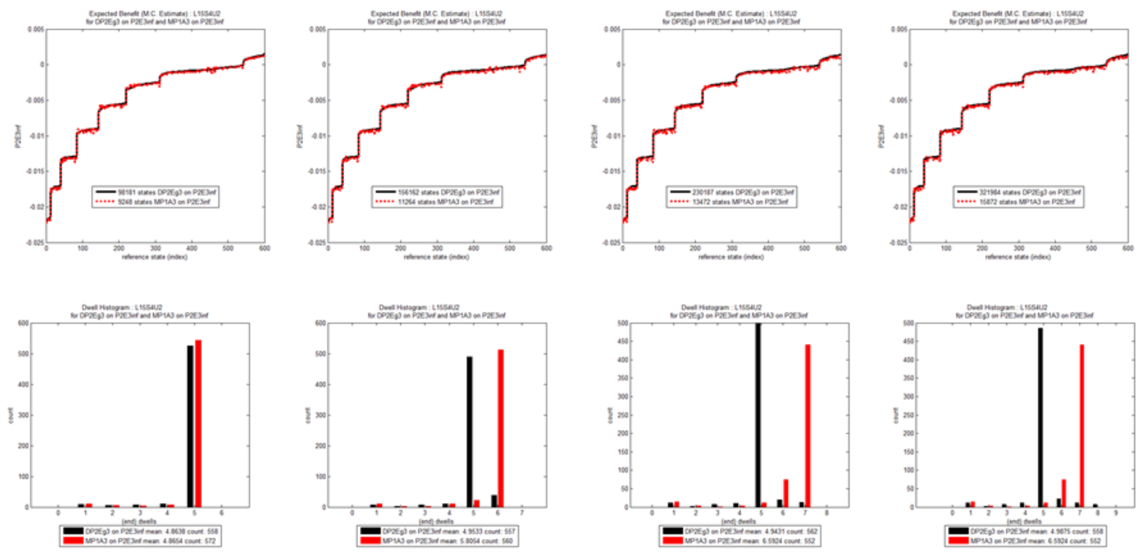


Figure 8.15 Performance Stable with Dwell Behavior Changing Maximum Dwell and Wait Increasing left (d=5, w=3) to right (d=8, w=6) (top: Optimal Performance bottom: Dwells Histograms)

With the optimized parameters, the Max-Plus Probability Solution is compared against the Aggregated Solution in Figure 8.16, demonstrating comparable output.

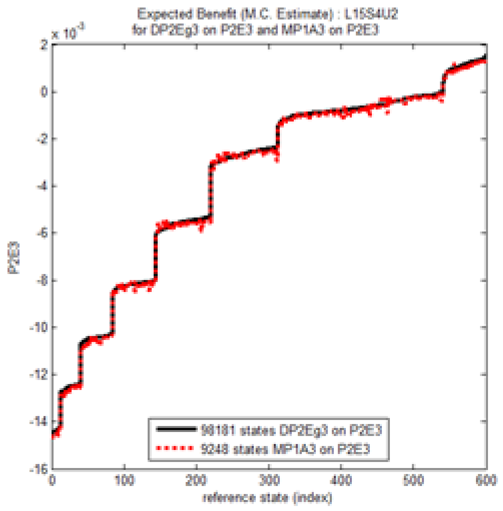


Figure 8.16 Max-Plus Probability Solution vs. Aggregate Solution

The Max-Plus Probability Solution dwell times are compared to the Aggregate Solution in Figure 8.l7.

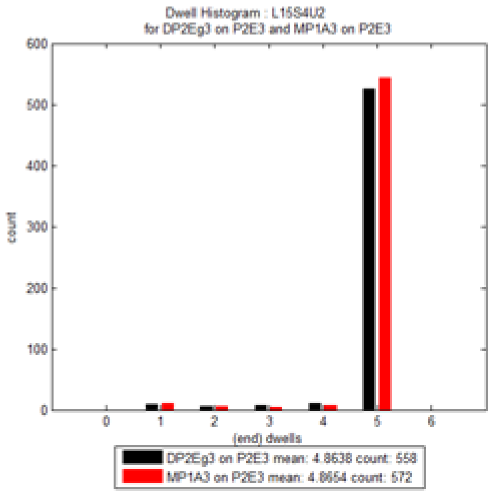


Figure 8.17 Max-Plus Probability Solution and Aggregate Solution Dwells

Likewise, Figure 8.18 compares the wait time distributions.

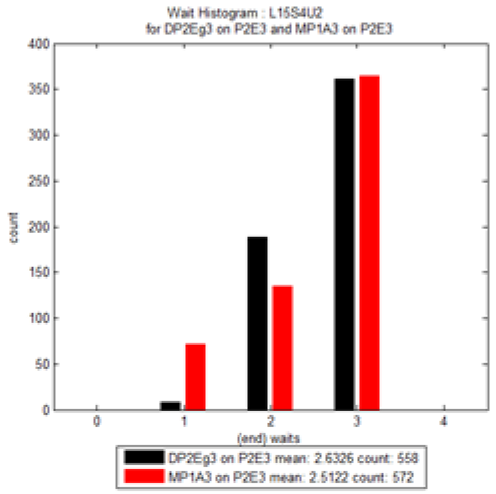


Figure 8.18 Max-Plus Probability Solution and Aggregate Solution Waits

8.4 Three UAVs

We now turn to a much larger problem, one with 30 discrete waypoint locations, 7 UGSs, and 3 UAVs. For such large size problem it is impractical to do much parametric testing directly. Since the baseline cannot be computed for the three-UAV problem, Figure 8.19 shows performance of only the Max-Plus Probability Solution. The third UAV is improving the maximum possible performance. Given its similarity for the single and dual UAV problems, this is likely close to optimal performance.

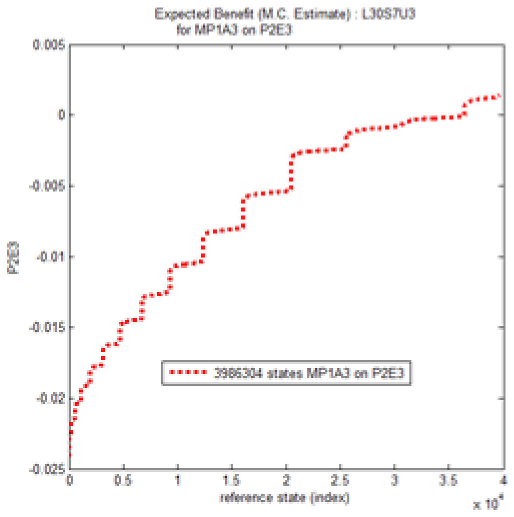


Figure 8.19 Max-Plus Probability Solution Performance

For validation purposes, we implemented a standard dynamic programming alternative to the reduced-state objective function on which our Max-Plus Probability Solution is based. Figure 8.20 shows the results of simulating both controllers.

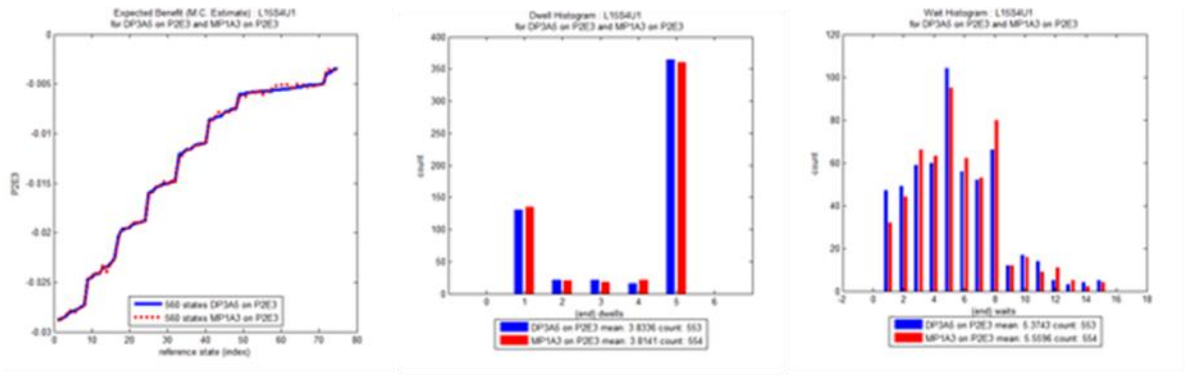


Figure 8.20 Comparison of Integration Methods

The performance of the alternative Max-Plus Probability Solution is comparable to that of the prior dynamic programming and state-aggregated dynamic programming approaches. The max-plus probability solution has the added benefit that it uses fewer states and can therefore solve larger problems than the standard and state-aggregated dynamic programming approaches. The max-plus approach leads to computational efficiencies for this example stochastic control problem.

9.0 <u>Conclusions</u>

We have developed a prototype software package, coupled with a hardware computational accelerator, to compute optimal controllers for general nonlinear problems. Our approach uses dynamic programming as the basic setting for optimization. Max-plus linearity of the Bellman equation of dynamic programming leads to an algebraic computational technique that permits efficient computation. Further benefits can be obtained by using FPGA hardware to build max-plus circuits.

We have also designed and developed a nonlinear controller for demanding perimeter patrol problems of interest to AFRL.

10.0 References

[AHHMF] Adams, M., W. Hall, M. Hanson, W. McEneaney, and B. Fitzpatrick, “Mixed Initiative Planning and Control under Uncertainty,” *2002 AIAA Proceedings on Unmanned Air Vehicles*.

[A] M. Akian, ``Densities of idempotent measures and large deviations," *Trans AMS* 351(11), pp 4515-4543, 1999.

[AGL1] Akian, M., S. Gaubert and A. Lakhoua. “The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis,” *SIAM J. Control and Optim.*, **47** (2008), 817–848.

[AGL2] Akian, M., S. Gaubert and A. Lakhoua. “A max-plus finite element method for solving finite horizon determinsitic optimal control problems,” *Proc. 16th International Symposium on Mathematical Theory of Networks and Systems* (2004).

[AQV] M. Akian, J.P. Quadrat, and M. Viot. ``Duality between probability and optimization," in J. Gunawardena (Editor), *Idempotency*, Cambridge University Press, Cambridge, 1998.

[BCOQ] Baccelli, F.L., G. Cohen, G.J. Olsder and J.-P. Quadrat, *Synchronization and Linearity*, John Wiley, New York (1992).

[BJ1] Barles, G., and E. R. Jakobsen, “On The Convergence Rate of Approximation Schemes for Hamilton-Jacobi-Bellman Equations,” *Math. Modeling and Num. Anal.*, **36**, No. 1, (2002), 33-54.

[BJ2] Barles, G., and E. R. Jakobsen, “Error Bounds for Monotone Approximation Schemes for Hamilton-Jacobi-Bellman Equations,” *Preprint*.

[CHHDP] P. Chandler, J. Hansen, R. Holsapple, S. Darbha, and M. Pachter, ``Optimal Perimeter Patrol Alert Servicing with Poisson Arrival Rate," *Proc AIAA GNC*, Chicago, IL, 2009.

[CGQ] Cohen, G., S. Gaubert, and J.-P. Quadrat. “Duality and separation theorem in dempotent semimodules” *Linear Algebra and Appl.*, **379**, (2004), 395–422

[DS2047] Michael Donley and Norton Schwartz, ``United States Air Force Unmanned Aircraft Systems Flight Plan, 2009-2047. Headquarters, United States Air Force, Washington DC, 2009.

[F] Fitzpatrick, B. “Idempotent Methods for Control of Diffusions,” *Proc 2010 MTNS*, Budapest, July 2010.

[FLW] Fitzpatrick, B., Li Liu, and Yun Wang, “The Legendre Transform and Max-Plus Finite Elements, *Proc 2011 Chinese Conference on Decision and Control*, pp. 2088 – 2092.

[FM1] Fitzpatrick, B. and W. McEneaney, “Control for UAV Operations under Imperfect Information,” by *2002 AIAA Proceedings on Unmanned Air Vehicles*.

[FKS] Wendell Fleming, Hidehiro Kaise, and Shuenn-Jyi Sheu, ``Max-Plus Stochastic Control and Risk-Sensitivity," *Appl Math Opt* 62(1), pp. 81-144, 2010.

[FM2] Fleming, W. H. and W. McEneaney. “A Max-Plus Based Algorithm for an HJB Equation of Nonlinear Filtering,” *SIAM J. Control Opt.* 38 (2000), pp 683-710.

[FS] Fleming, W. H., and M. Soner. *Controlled Markov Processes and Viscosity Solutions,* Springer-Verlag, New York (1993)

[GRCF] Gross D, Rasmussen S, Chandler P, Feitshans G. Cooperative Operations in UrbaN TERrain (COUNTER). *Defense and Security Symposium*. SPIE: Orlando, FL,

[HOW] Heidergott, B., G. Olsder, and J. van der Woude. *Max Plus at Work*, Princeton University Press, Princeton (2006).

[JZ] Jones, J., and F. Zufryden. "Adding Explanatory Variables to a Consumer Purchase Behavior Model: An Exploratory Study," *Journal of Marketing Research* **17**, 323-34, (1980).

[Ke1] Ronald Kessel, ``The Burden of Computer Advice: Using Expert Systems as Decision Aids," *DRDC Atlantic Technical Report TR 2003-241*, Defence R & D Canada, 2003.

[KM] Kolokoltsov, V.N., and V.P. Maslov. *Idempotent Analysis and Its Applications,* Kluwer (1997).

[KPDC] Krishnamoorthy Kalyanam, Meir Pachter, Swaroop Darbha and Phil Chandler, "Approximate dynamic programming with state aggregation applied to UAV perimeter patrol," *Intl J. Robust and Nonlinear Control*, v 21, no 12, pp 1396-1409, 2011.

[L] Litvinov, G. L. "The Maslov Dequantization, Idempotent and Tropical Mathematics: A Very Brief Introduction," ESI Preprint 1588, (2005).

[LMS1] Litvinov, G.L., V.P. Maslov and G.B. Shpiz, "Linear Functionals on Idempotent Spaces: An Algebraic Approach," *Dolkady Math*, **58**, No. 3 (1998), 389-391.

[LMS2] Litvinov, G.L., V.P. Maslov and G.B. Shpiz, "Idempotent Functional Analysis: An Algebraic Approach," *Math. Zametki*, **69**, No. 5 (2001), 758-797.

[Mc1] McEneaney, W. M. "Distributed Dynamic Programming for Discrete-Time Stochastic Control and Idempotent Algorithms," to appear (2008).

[Mc2] McEneaney, W. M., *Max-Plus Methods for Nonlinear Control and Estimation*, Birkhauser, Boston, 2006.

[MC] W.M. McEneaney and C.D. Charalambous, ``Large Deviations Theory, Induced Log-Plus and Max-Plus Measures and their Applications" in *Proc MTNS 2000* Perpignan, France, June 19-23, 2000.

[MDG] McEneaney, W. M., A. Deshpande, S. Gaubert. "Curse-of-Complexity Attenuation in the Curse-of-Dimensionality-Free Method for HJB PDEs," *Proc. ACC* (2008), 4684-4690.

[MOC1] McEneaney, W. M., Ali Oran and Andrew Cavender, "Value-Based Control of the Observation-Decision Process," *Proc. ACC,* (2008), 5041-5046

[MOC2] McEneaney, W. M., Ali Oran and Andrew Cavender, "Value-Based Tasking Controllers for Sensing Assets," *Proc AIAA GNC* (2008).